2005

Athena: An online proposal development system

Humaira Rahim

ATHENA

AN ONLINE PROPOSAL DEVELOPMENT SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Humaira Rahim

March 2005

ATHENA

AN ONLINE PROPOSAL DEVELOPMENT SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Humaira Rahim

March 2005

Approved by:

[redacted] 12/12/05

Dr. Richard Botting, Chair, Computer Science Date

[redacted]

Dr. Arturo Concepcion

[redacted]

Dr. Ernesto Gomez

[redacted]

Dr. David Turner

ABSTRACT

Digitizing the proposal development process is a dream of every educational system or Collaborative Application Developing agencies. And making this dream a reality is a great challenge.

Athena - Online Proposal Development System was the first version of a vision of Dr. Richard Botting, Professor, Department of Computer Science, California State University, San Bernardino towards digitizing the writing, review and annotation of project proposals.

Athena was an effort to help the Computer Science Master's students in providing their project proposals online for review and annotation by the committee members. It was also designed for the faculty members to manage the project proposals of students they are advising or are guiding as a committee member.

Athena aimed at facilitating communication between the student and the committee as well as minimizing the re-currency of comments by different committee members, reducing the calendar time and amount of paper used in the repetitive review and updating processes of proposal development. Athena caters to advancing Master's students to Candidacy in a faster yet efficient manner.

ACKNOWLEDGMENTS

First and foremost I thank God Almighty for all His blessings. I especially thank my parents in having faith in me and instilling in me a strength of mind and character that helped me in facing every challenge with confidence and poise.

I would like to thank my Aunt, Rafath, and her husband, Muqtadir. It was their guidance, support and encouragement throughout that has helped me in pursuing my Masters and living in America. I thank my Aunts, Uncles, my brothers and sisters, my husband and his family, and the rest of my family for all their love and support. It is because of them that I had the strength to face each new day with a smile.

Special thanks to my Advisor Dr. Richard Botting. He has been my best critic and my ever present source of encouragement. It is because of his experience, constant guidance and faith in my abilities that I have been able to face challenges in this field as well as in implementing this project.

I would like to thank Dr. Concepcion, Dr. Gomez and Dr. Turner for their guidance in my project and all through my Masters. It was Dr. Concepcion who taught me how good communication skills, the ability to work in a

team and the application of parallel processing in real life are important to succeed and not very difficult for an individual to master. Dr. Turner, I thank him for giving me the real life scenario of where the Web programming world stands and challenging me and guiding me to work towards achieving that level. If not for him I would not have the courage to learn a new programming technique for this project. I thank Dr. Gomez for always having answers for my questions or helping me find it.

I would also like to thank all the Professors of the Department for guiding me in becoming a computer professional. I thank Ken Han and Monica Gonzales for all their encouragement and help throughout my Masters. I also thank my friends for their encouragement and support.

Finally I would like to acknowledge the support of The National Science Foundation grant #9810708.

DEDICATION

To the dynamic women in my life, my Grandmothers Ashrafunnisa and Naseerunnisa, my mother Ayesha, my aunts Layeeq and Rafath, and my sisters Hajira and Maria. You are my inspiration and strength.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction

The Internet with its continuous development and popularity has brought about a wave of optimism and interest that attracts everybody to it. The educational system has benefited tremendously from it. Information about anything is at the tip of your fingers. Distance is no barrier anymore. Communication has taken a new meaning.

The process of writing proposals in the educational field involves a series of scheduled meetings, scribbled and illegible notes, tons of paper and repetitive annotations by reviewers. There have been various solutions to this problem, but none so versatile that it could cover every possible detail of proposal development. Proposal development differs in every field of study and has various rules and regulations depending on the type of department.

Keeping in mind both these scenarios, Athena - An Online Proposal Development System has been developed for the Department of Computer Science at California State University San Bernardino.

1.2 Purpose of the Project

The purpose of the project was to develop an online system for the Department of Computer Science at California State University San Bernardino that could help Masters' students and their project committee members in composing, reviewing and annotating proposals.

1.3 Scope of the Project

Athena utilizes the new ability of professors and students to work comfortably on the Internet, thus bridging the communication gap as well as providing an efficient mechanism to compose, review and annotate proposals.

This first prototype caters only to Masters Students working on their project proposals with a committee of faculty members.

Each committee consists of one advisor and two committee members for the reviewing process. The current procedure has the following steps:

- Student turns in a version of the proposal to an advisor who first reviews it and suggests corrections and changes;
- Student turns it in again to the rest of the committee members and they review it.

- After several repetitions of this process the proposal is finally approved and the student advances to Candidacy.

The issues involved are scheduling problems when students need to meet with their committee members, repetitive annotations because the committee members review it at the same time, wastage of paper and time.

Athena tried to handle some of these issues. It had an environment that enabled students to create their proposals online and make it available for committee members to annotate. It enables the committee members to manage and review proposals of students they are advising or guiding.

Athena had three types of users:

- System Administrator
- Student
- Faculty Member

The use case diagrams showing the individual functions of all the users are given in the following figures:

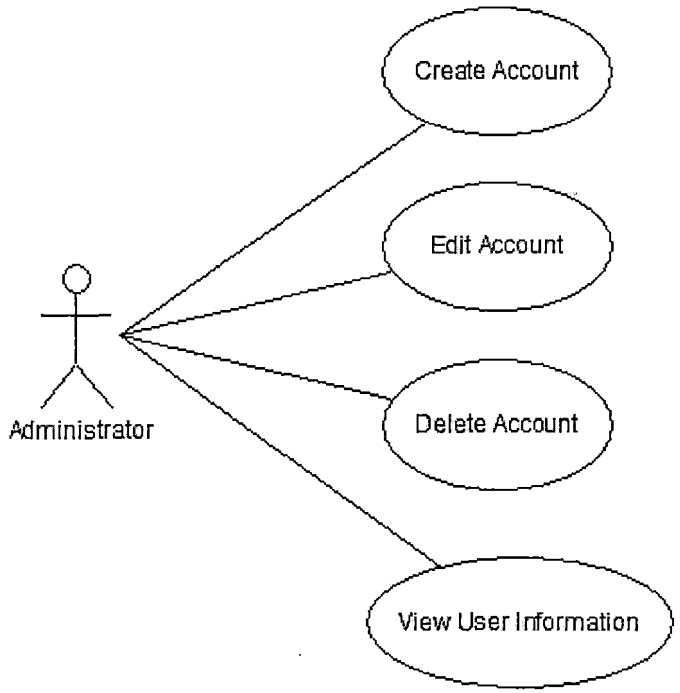

Figure 1. Administrator Use Case

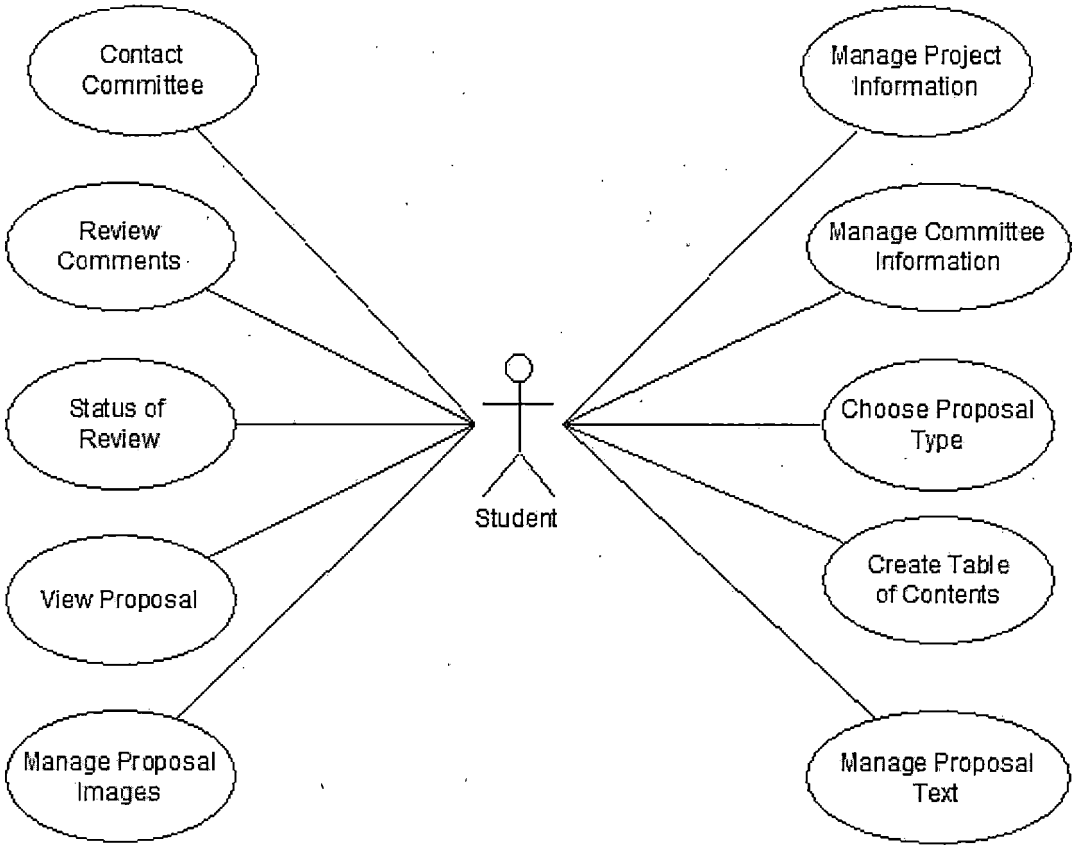

Figure 2. Student Use Case

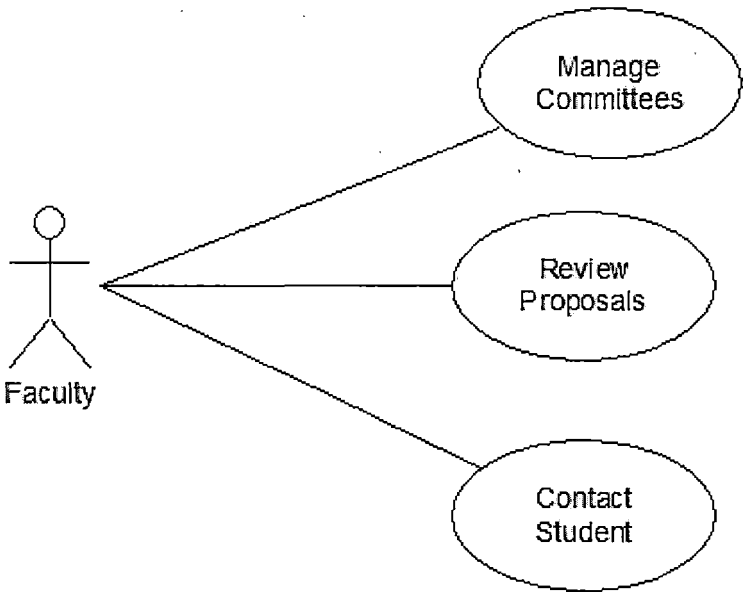

Figure 3. Faculty Use Case

1.4 Context of the Problem

The context of the problem was to address issues of communication, readability, repetitive annotations, varied locations of students and faculty, wasting resources used during the Masters Project Proposal Review.

1.5 Related Work

There has been lot of work in the field of digital thesis preparation and collaborative development of documents. The work on Athena has been preceded by research and projects of similar kind. The Universidad de Las Americas, Puebla(UDLA) has under-taken various

projects under an ambitious digital libraries program that they have termed University Digital Libraries for All (U-DL-A). U-DL-A is available online at http://biblio.udlap.mx/tesis.

There have been numerous projects that they have developed as part of the program in conjunction with Virginia Polytechnic Institute and State University, Blacksburg, USA. The projects developed under U-DL-A that Athena has taken ideas from are Tales [1], Zeus [2] and Poseidon [3].

The Tales [1] project started in 1999 is currently a university-wide deployment that constructs digital collections in Spanish and serves as a platform for research of open issues in digital libraries. The features of Tales can be summarized as following:

1. Information of students, courses and professors is entered into the databases and Academic departments or thesis coordinators define the composition of thesis committees.
2. Students submit their documents and advisors approve the document for publication.
3. The file format in which students submit documents are PDF and MS word. The file format used to save

documents is HTML. Displaying documents is through HTML or PDF.

4. There is a fixed structure to the thesis which is based on the data type definition for Electronic Theses and Dissertations (ETD-ML). Only those elements relevant to the universities thesis regulations were selected.
5. Once the thesis is complete it becomes part of a collection and its contents become accessible to users as specified by the thesis advisor
6. Access to the thesis is through interfaces for searching and browsing.
7. All the tools were developed using Java, Java Servlets and MySql database.

The Zeus [2] project is another tool developed under the U-DL-A initiative that is a web based cooperative environment designed specifically to support the process of reviewing, annotating and publishing a thesis in a digital library.

The features of Zeus can be summarized as follows:

1. Zeus has three main components that it implements i.e. Composition, Annotation and Visualization and three possible roles for users i.e. student, advisor and member.

2. The composition tool helps the student upload the thesis into Zeus' workspace. The file format for upload is HTML.
3. The annotation tool is used by advisors and members to make comments and annotations to the document. The annotation tool is highly customizable. There are colors assigned for buttons that symbolize the most commonly used annotations i.e. arbitrary comments, replace existing text, insert text, delete text, and spell check. These annotations are assigned to the text of the thesis just by selecting the part of the document to which it applies. This leaves the document clear and legible with the text in different colors.
4. The Visualization tool displays the annotations and comments that are "hidden", linked to their corresponding text. They are shown in different frames depending on the number of reviewers the student selects. This tool also enables the reviewers to check if annotations were applied to the document.
5. Zeus in its current implementation handles version control in a simple manner. After

turning in a thesis to a reviewer a new version cannot be uploaded until the reviewers explicitly release the current document after annotating it. After final approval Zeus automatically incorporates the thesis into the digital library.

6. Zeus was implemented using Java and the Swing API with Informix Universal Server database management system.

The Poseidon [3] is a software component for U-DL-A that goes a step further than Tales and Zeus and focuses on collaborative revision of writing projects in general and provides these following features:

1. Provides the facility for users to define the document types (books, journals and conference papers, internal reports, theses and letters etc.) as well as their own formatting requirements.
2. Users can establish roles and annotation conventions. Annotations are color coded and are user defined. Portions of text are highlighted wherever comments apply.

3. Simultaneous annotation of documents from remote sites using conventional Web browsers is facilitated.
4. Version control is maintained by having a public version of the document that is "read-only" and another version which is available for review by the assigned members. If two reviewers log in at the same time, one gets a "read-only" and the other the annotatable version. Once the review is done the document is available for the other reviewers. Once approved the document is added to the digital library.
5. Poseidon was developed using Java JDK 1.4.

Apart from the U-DL-A projects there have been various other related works including [4].

1.6 Significance of the Project

Athena was developed for the Masters students' and their committees to compose, review and annotate project proposals over the Internet thus reducing the calendar time involved in approval of proposals, reducing the repetitive comments by the committee, reducing the amount of paper used during this process and providing a readable, efficient and friendly environment. Athena could

also be used by Software Engineers who are following the IEEE Std. for Software Requirement Specifications [5].

1.7 Assumptions

The following assumptions were made regarding the project:

1. The users understand the basic rules and regulations governing the project proposal approval system.
2. There are three types of users: Administrator, Student and Faculty. Each of them has their respective rights. Users of Athena do not assign rights and permissions on documents or other users; it implements it through the rules of the Department regarding this issue.
3. The Administrator needs to be contacted for the first time a student wants to use the system for proposal development and review. Once the student is added to the database, a username and password are assigned and the student can login with that.
4. The Administrator enters the information of all the faculty members of the Department into the database. The faculty members will have their

own username and password to access the system. These are provided by the systems administrator.

5. For each project, the students will form their own committees consisting of one advisor and two members for the review, choosing from the list of faculty provided by Athena.
6. The student composes the proposal in the Athena workspace over the Internet.
7. The communication between the faculty and student is through emails generated in Athena. An external email system need not be used.
8. The number of times the review will be done depends on the student and the committee.
9. The comments table has the provision for the students to add/delete/update comments as a reply to annotations made by the committee members.
10. How they wish to proceed to get the approvals of the committee depend upon their mutual consent which is taken via email.
11. Advisors have the right to delete a proposal from the database at a later stage if they wish to. The student gives up this right after the approval is done.

12. This way the only formally scheduled meeting with the faculty would be when the student needs to take formal signatures of the committee after the final approval of the proposal is done via email to advance the student to Candidacy.

1.8 Limitations

During the development of the project, a number of limitations were noted. These limitations are presented here:

1. Each User can have only one role.
2. There can be only 1 image per section. The formats of images accepted in Athena are: JPEG, GIF, BITMAP, and PNG.
3. The student can choose from only two ways of creating a proposal i.e. Strict IEEE std. based Proposal and IEEE based yet User-Defined Proposal.
4. In the Strict IEEE std. based proposal the Table of Contents has very limited flexibility. The user cannot change the headings or add/delete the existing sections of the standard. If a particular section does not apply to the proposal

then a "NOT APPLICABLE" comment can be added for that section.

5. The only flexibility provided is that the student can add sub-sections to the sections: 2.3 User Characteristics and 3.1 External Interfaces. This is allowed since these are the two sections where the images will be more than 1. Since User characteristics have use case diagrams of users and External Interfaces have the interfaces of the project, both of which are multiple and would require multiple sections to hold the images.
6. The image limitation on the proposal creates the only flexibility in the Strict IEEE Std. based Table of Contents, that is managing a third level for the above mentioned sections. The Limitations in these are that they cannot have sub-sub-sections, though they can update the headings or be deleted.
7. The other type of proposal creation is the IEEE Std. based yet User-defined proposal. In this case the limitations are applied based on a survey conducted on the Proposals of graduated Masters Students. [Table 1]

8. The table of contents of the User-defined proposal can have only three main chapters and none of them can be deleted. There cannot be sub-sections of sub-sections in the table of contents. The titles of all sections and chapters can be edited and updated at any time during proposal creation.
9. A text area cannot be added.
10. Previous versions of proposals are not maintained.
11. The Advisor and Committee Members cannot modify the proposal. They can only read and add comments.
12. There is no automatically generated communication.
13. The advisor is the only committee member who has the right to delete a proposal from the database.
14. There cannot be any HTML tags in the text.
15. The students can add/update/delete the reply they made to a faculty comment. They cannot add comments to a section that does not have any comments by committee members.

1.9 Definition of Terms

The following terms are defined as they apply to the project:

1. Java - An object oriented language developed by Sun Microsystems. Java programs are capable of running on most popular platforms without the need for recompilations.
2. JDBC - Java DataBase Connectivity. A programming interface that lets Java applications access a database via the SQl Language.
3. JSP - JavaServer Page, an extension to the Java servlet technology from Sun Microsystems that provides a simple programming vehicle for displaying dynamic content on a web page.
4. JavaBean - A component architecture for the Java programming language, developed initially by Sun, but now available from several other vendors. JavaBeans components are called "Beans".
5. JavaScript - A Scripting Language that is widely supported in the web browsers and other web tools. It adds interactive functions to HTML pages which are otherwise static.
6. JDK - Java Development Kit, A free Sun Microsystems product which provides the

environment required for programming in Java. The JDK is available for variety of platforms, such as Sun Solaris, Microsoft Windows and Linux.

7. J2EE - Java 2 Platform, Enterprise Edition. Sun's Java platform for multi-tier server oriented enterprise applications.
8. OS - Operating System. The low-level software which handles the interface to peripheral hardware, schedules tasks, allocates storage, and presents a default interface to the user when no application program is running.
9. JVM - Java Virtual Machine. It is a software that interprets and executes the byte code in Java class files.
10. SQL - Structured Query Language. A standard language that provided controlled access to databases.
11. Browser - A program capable if retrieving HTML documents that includes references to images and Java byte code and rendering it into a user-readable document.
12. MPDS - Masters Proposal Development System. This is the name of the directory where the .jsp files

of Athena are stored in tomcat. The same name has been used for the database of Athena as well.

1.10 Organization of the Project Document

The Masters Project document was divided into six chapters. Chapter One provides software requirements specification, an introduction to the context of the problem, purpose of the project, related works, significance of the project, limitations, and definitions of terms. Chapter Two consists of the software design. Chapter Three documents the steps used in testing the project. Chapter Four presents the maintenance required from the project. Chapter Five presents the users manual from the project. Chapter Six presents conclusions drawn from the development of the project and future developments. The Appendices containing the project's database schema and source code follows Chapter Six. Finally, the references for the project are presented.

CHAPTER TWO

SOFTWARE DESIGN

2.1 Introduction

Chapter Two consists of a discussion of the software design. Specifically, Athena was a JSP based system that resided on a three tier architecture. The front-end was based on JSP, and the back-end was a MySql database. The connections to the database were made through the java beans with a JDBC connection.

2.2 Preliminary Design

Athena was a system designed for the Department of Computer Science at California State University San Bernardino to help students and faculty to compose, review and annotate proposals. It was developed to overcome the following issues:

1. Scheduling meetings between students and committee members each time the student wanted to get a proposal reviewed.
2. Usually the students pursuing their masters are working either full time or part time have issues that don't really facilitate their visits to the campus.

3. The annotations made on the paper were sometimes illegible and spanned multiple pages.
4. The same comments were made by different committee members as they could not see the comments made by the others.
5. Lots of paper was wasted because of the multiple times the proposal was edited based on the comments of the committee.

A robust, flexible and user-friendly system was needed and Athena was an implementation that solved most of the issues.

2.3 Detailed Design

Logic

All the user interfaces were divided based on user type.

For the user Administrator the use cases are:

1. Create User Account: The system administrator enters the user information and stores it in the database.
2. Edit User Account: This use case has two scenarios: Scenario A: List all users: The administrator chooses the user to edit from the list of users. Scenario B: Edit user

information: The system brings up that particular chosen users information for the administrator to edit.

3. Delete User Account: The administrator can click on a hyperlink and delete a user from the list of users displayed by the system.
4. View User Information: The administrator can view the information of all the users of the system.

For the user Student the use cases are:

1. Manage Project Information: This use case has three scenarios. Two of them are based on the fact that the user has entered the project information or not. Scenario 1: Add Project Information - If this is the first time the user has logged-in, the system will take the user to add Project Information. Scenario 2: View Project Information - If the user is has already entered the project information and is coming back to just view it, this scenario takes place. It has the option of the third scenario which is Update Information. Scenario 3: Update Project Information - The user can update project information as needed.

2. Manage Committee Information: This use case has 2 scenarios. Scenario 1: Choose Committee - The student can choose from the list of faculty members in the system those whom they want on their committee. Scenario 2: Update Committee - The student can update the committee that it had entered before. All the committee members status is set to Not Approved until they are approved by the Advisor of the Proposal Committee.
3. Choose Proposal Type: The student can choose between a Strict IEEE Std. [5] based Proposal and a IEEE std. based yet User-defined Proposal. The features are explained in the following sections.
4. Create Table of Contents: The student creates the table of contents based on the type of proposal they choose.

   The features of the Strict IEEE Std. Based Proposal are:

   a. Follows the IEEE Std. 830 - 1998 [5] STRICTLY.
   b. The student cannot add sections, delete sections or update headings of any of the existing sections.

c. The sections where the student does not have any content they could just say "Not Applicable to this project".

d. The only flexibility provided is in the sections 2.3 User Characteristics & 3.1 External Interfaces of the standard.

e. This flexibility is provided because these sections will have multiple images and because of the limitations of the project each section can have only one image.

f. The user can add/delete/change headings of sub-sections to ONLY these two sections (2.3 User Characteristics & 3.1 External Interfaces).

The features of the user-defined proposal are as follows:

a. It is based on the IEEE std. [5] but has a lot of flexibility.

b. The user cannot delete or update the 3 chapters nor can they add sub-sections at the fourth level.

c. The table of contents is limited to 3 levels only. This conclusion has been drawn from a survey conducted whose results are summarized later in this section.

d. The user can add sections, add subsections, delete sections, delete subsections, and update headings of all the sections except the special ones mentioned above.

Once the user creates the tables of contents then the final version is shown to the user for approval. It can be changed at any point.

5. Manage Proposal Text: This use case has 2 scenarios. Scenario 1: Add Text - The system lets the user add the text by chapters. Scenario 2: View Text - Once the student has saved the data it can view it.
6. Manage Proposal Images: This use case has 3 scenarios. Scenario 1: Add Images - Once the student has added the text it can add images where needed. Scenario 2: View Image - After the user chooses the images and uploads it the user can again see the proposal with both the text as well as the image. Scenario 3: Delete Image - The student can delete the image at the view proposal by chapter's stage.
7. View Proposal: The student can see the complete proposal with the images and text all in one page

and not by chapters. It cannot make any changes here.

8. Status of Review: The student can check the status of proposal review by the committee members and depending on it can go and check their comments.
9. The status could be: Not Reviewed, Reviewed & Approved Reviewed & Commented, and Commented & Approved.
10. Review Comments: The Student reviews the comments made by the faculty and add their own reply to it. The student cannot add comments to a section. They only review comments and give a reply to an existing comment. The scenarios are: Scenario 1: View Comments - View the comments by the committee members. Scenario 2: Add Comment - Add their own reply to this committee comment. Scenario 3: Update Comment - Update their own comments. Scenario 4: Delete Comment - Set its own comment to none again, if you do not want to reply to a comment made by a committee member.
11. Contact Committee Members - This is where the student can email the committee members regarding the proposal.

For the user Faculty the use cases are:

1. Manage Committees - The faculty members can view the proposal committee's they are part of divided based on whether they are the advisor or a committee member. There are 2 scenarios. Scenario 1: New Committee - The faculty when playing the role of advisor need to approve the status of the committee members for them to be able to view proposals. Scenario 2: View Committee - If the advisor has approved the committee then they see the committee members with their individual status. As committee members the faculty can view the committee of a particular project at any time.
2. Review Proposals - The faculty member can choose from the proposals which one they want to review. There are 4 scenarios here. Scenario 1. Review Proposal - The whole proposal is displayed along with links to add or view comments for each section. Scenario 2: Add Comment - The faculty can add 1 - many comments for each section. Scenario 3: View Comment - The faculty can view the comments already made for a particular section even before they add their own comments

thus not repeating a comment. It is recommended that the faculty does this to avoid repetition of comments as well as saving time. Scenario 4: Delete Comment - The faculty can ONLY delete a comment that they made themselves. They cannot delete any other comments made by the other committee members.

3. Contact Student - The faculty can contact any student by using Athena's email system.

These are all the use cases of the users of Athena. As mentioned above the table of contents functionality for the User-defined type of proposal has been implemented because of a survey conducted. To have a uniform structure of the table of contents for the system a survey was conducted. 12 project proposals of Masters Students of Computer Science were surveyed. These were the findings:

1. All of them followed the IEEE Guidelines for Software Requirement Specifications [5].
2. 9/12 proposals had a maximum of 3rd level headings.
3. All 12 proposals had an Introduction.
4. 9/12 proposals named/had similar chapter as Overall Description.

5. 7/12 proposals had the third chapter as Specific Requirements.
6. 9/12 students had 3 headings at Level 1(Chapters Level).
7. 3-4/12 students had about 4-6 headings at Level 2 (Sections Level).
8. 6/12 students had at-least 3 headings at Level 3 (Subsections Level).
9. All the other levels were ignored as it was already deduced that the majority of students went up to a level of 3 chapters.

Depending upon all these factors as well as the IEEE std. [5] Athena's Table of Contents structure was decided. The following table shows the results of the survey:

Table 1. Survey Conducted for Table of Contents

Student	# of Levels	Depth at Level 1	Depth at Level 2	Depth at Level 3	Depth at Level 4	Depth at Level 5
1.	3	3	6	3	-	-
2.	4	3	4	3	3	-
3.	3	4	5	3	-	-
4.	3	3	4	4	-	-
5.	3	3	4	3	-	-
6.	5	3	5	6	3	3
7.	3	3	8	5	-	-
8.	2	3	5	-	-	-
9.	3	4	4	3	-	-
10.	3	2	5	3	-	-
11.	3	3	6	5	-	-
12.	3	3	6	8	-	-

The other type did not require any survey as it follows the IEEE standard [5] strictly. This was how the logic of Athena developed.

Architecture

Athena had 3-tier architecture. The 3-tier architecture was considered since the 2-tier architecture combines the presentation logic with the business logic in one tier called the client side. The other tier was called the server provides the database.

The 3-tier architecture separates the business logic from the presentation logic and has the database in the third tier. This architecture is very flexible and has high scalability.

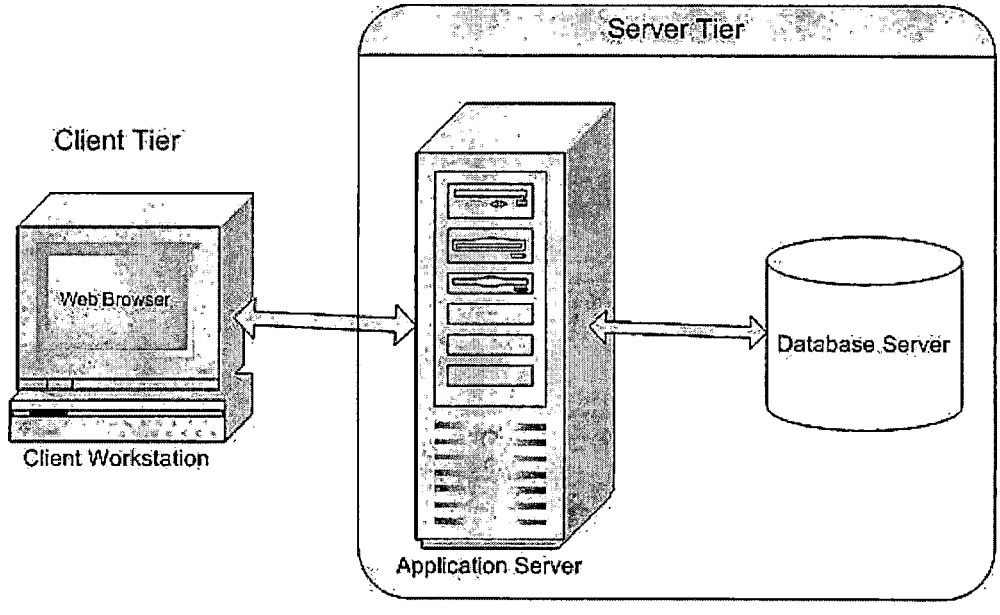

Figure 4. 3-Tier Architecture Diagram

Client Tier. The Java enabled web browsers are used as client tier. The user sends requests via the browser. The browser then sends the request to the Java Beans residing at the application server that process the request and send it back to the browser which in turn interprets the information it receives from the server and displays graphically for the client. The user calls methods of Java classes to incorporate the functionality. These calls are handled at the middle tier.

Middle Tier. The middle tier is also known as the application server. Athena was designed to run in Jakarta Tomcat-5.0.30 web server. Tomcat supports JSP and JavaBeans which are the programming techniques used in Athena. The Java classes that are used to perform the functionality and do the interaction with the database in this tier are:

- Database
- Users
- UsersAdmin
- Proposal
- StudentAdmin
- SectionAdmin
- CommentsAdmin

The class diagrams of these classes can be diagrammatically represented as follows:

1. Database.java: This class only has a method to connect to the database and returns a connection.

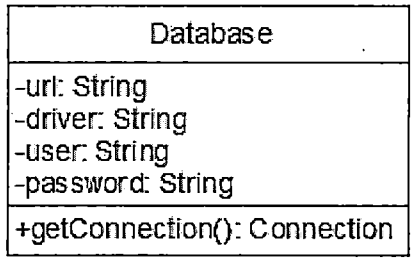

Figure 5. Database

2. Users.java: This class has the get() and set() methods of the users table in the database and this creates an object for the user bean.

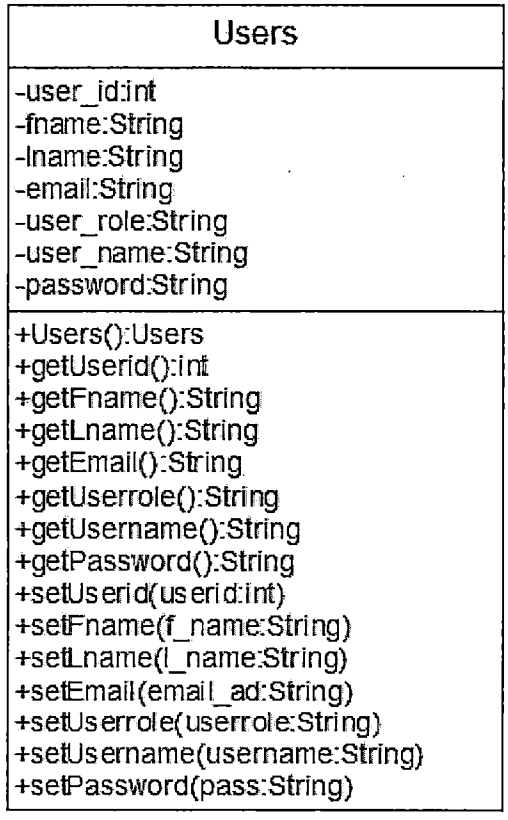

Figure 6. Users

3. Proposal.java: This class has the set() and get() methods for the proposal table in the database. This creates an object of type proposal to be used in the interfaces.

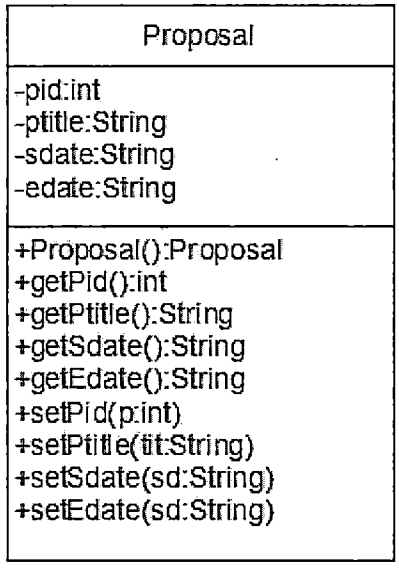

Figure 7. Proposal

4. SectionAdmin.java: This class has methods to create sections, delete sections, insert section and insert subsection.

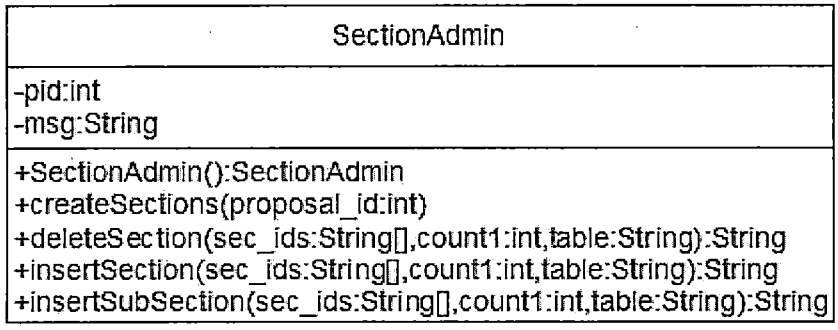

Figure 8. SectionAdmin

5. StudentAdmin.java: This class has methods to add proposal, modify proposal, create committee and update committee. These are related to the Project and Committee tables of the database and manage these functions of the student. In the interfaces these are the first two functions done which affect all the others that follow.

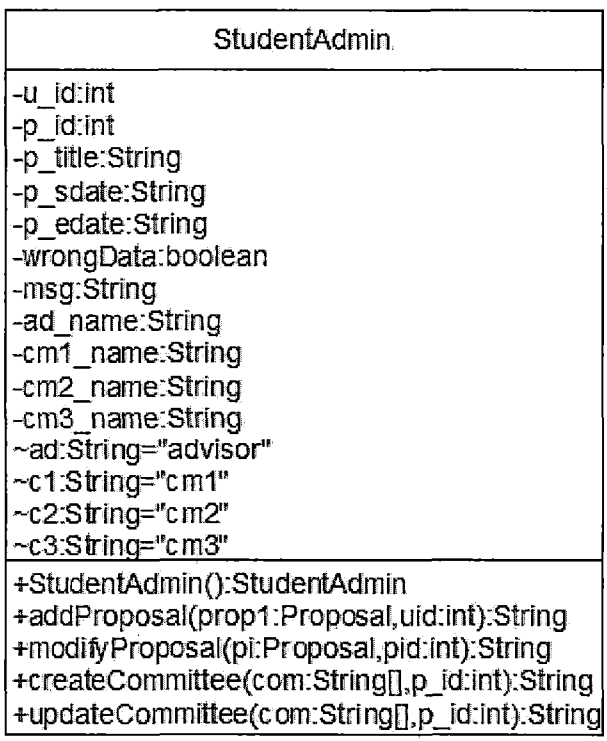

Figure 9. StudentAdmin

6. UsersAdmin.java: This class manages the user - systems administrator's functions of managing user information. It has methods to create user, delete user, edit user and login user. The Login user method validates the user entries to check if that user exists before allowing access to Athena.

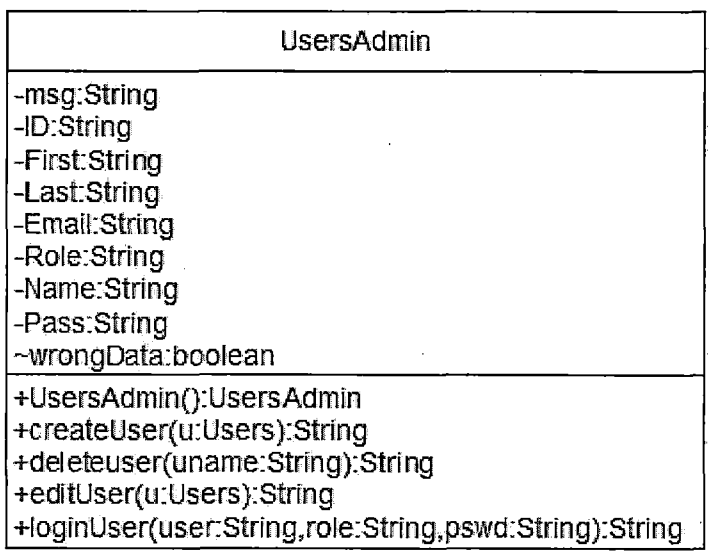

Figure 10. UsersAdmin

7. CommentsAdmin.java: This class has methods that take of the comments of the faculty as well as the students.

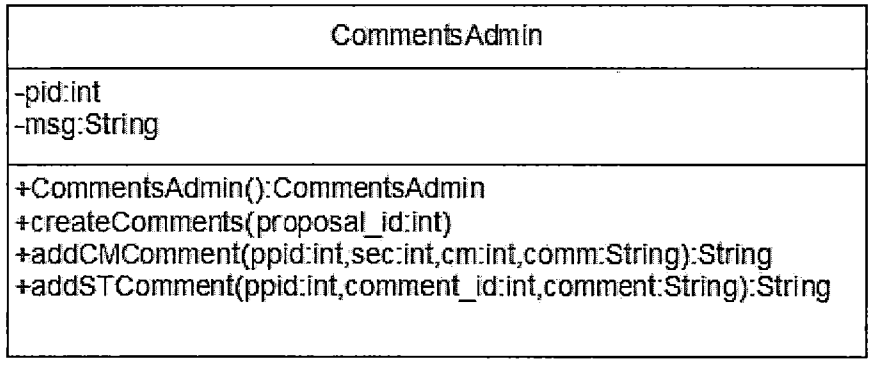

Figure 11. CommentsAdmin

The class diagram of Athena showing the relationship between the classes.

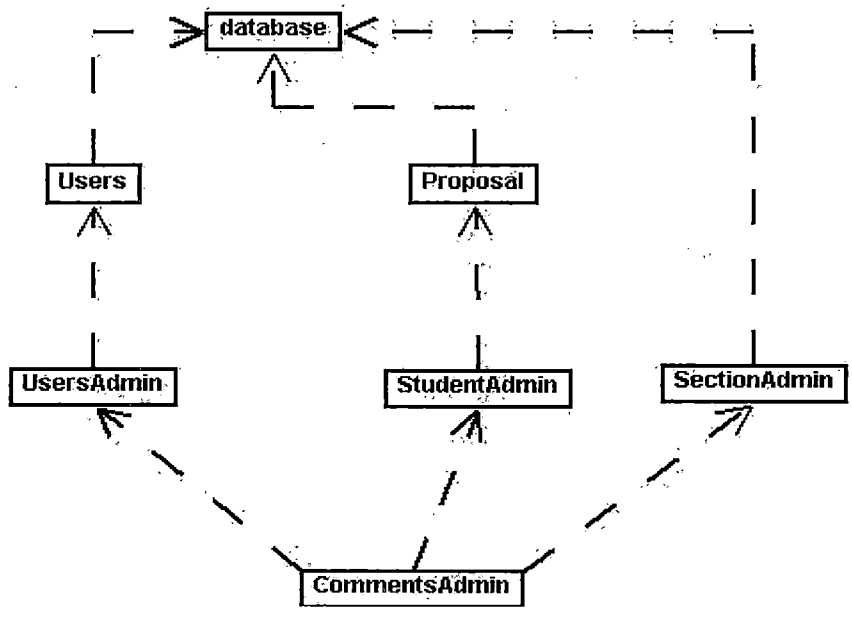

Figure 12. Class Diagram of Athena

Database Server Tier. The database tier is the back end application server where the data is stored. Athena uses MySql version 4.0.23-nt. MySql provides very fast joins using an optimized one-sweep multi-join. You can connect MySql to Tomcat using a MySql driver.

Programming Technique and Language

The programming language used by Athena was Java [8] and its JSP technology [9]. JSP standard was developed by Sun Microsystems as an alternative to Microsoft's active server page (ASP) technology. JSP pages are similar

to ASP pages in the fact that they are compiled on the server, rather than in a user's Web browser.

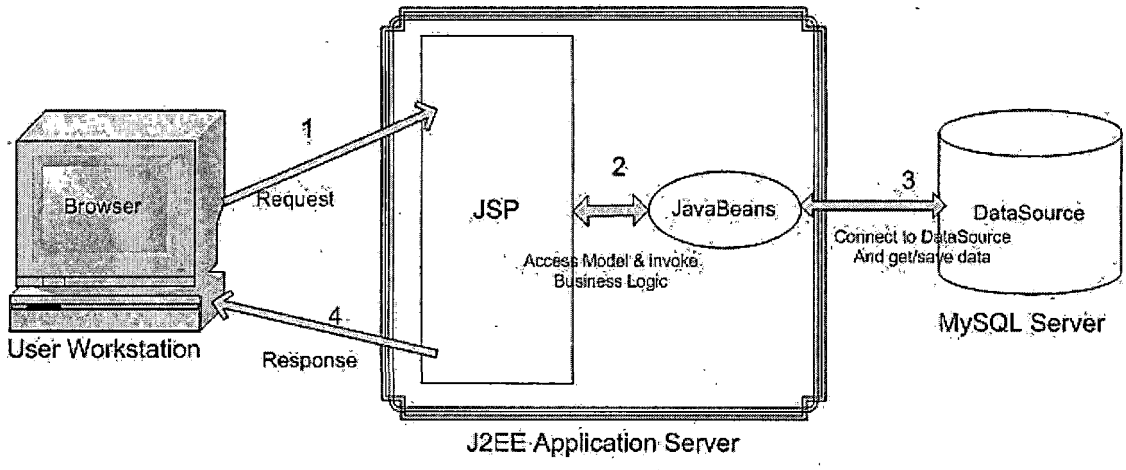

Figure 13. Logical Data Flow Diagram

However, JSP is Java-based, whereas ASP is Visual Basic-based. JSP pages are useful for building dynamic Web sites and accessing database information on a Web server. Though JSP pages may have Java interspersed with HTML, all the Java code is parsed on the server. Therefore, once the page gets to the browser, it is only HTML. JavaScript, on the other hand, is usually parsed by the Web browser, not the Web server.

The other benefits of JSP that made it a choice over other technologies are:

1. JSP separates program logic from the presentation.

2. JSP pages have the "Write Once, Run Anywhere" property. By virtue of their ultimate translation to Java byte code, JSP pages are platform independent. This means that JSP pages can be developed on any platform and deployed on any server.
3. JSP pages also make it easy to embed reusable components like JavaBeans that perform specialized tasks. JavaBeans are developed once and can be used in any number of JavaServer Pages.
4. JSP has custom tag libraries that make it highly extensible.

These numerous characteristics of JSP made it a good choice for Athena.

Database

The Database of Athena has been constructed based on the needs of the system. It has been normalized to the 3rd Normal form.

The tables created in Athena are:

1. Table users: This table stores the information of the users. This information is entered and maintained by the user - Systems Administrator. The table after the figure gives a description of each of the attributes of

the table users and what they store. There is testing at both the database level as well as the interface level by first Java Script and then JSP which catches the SQL exceptions.

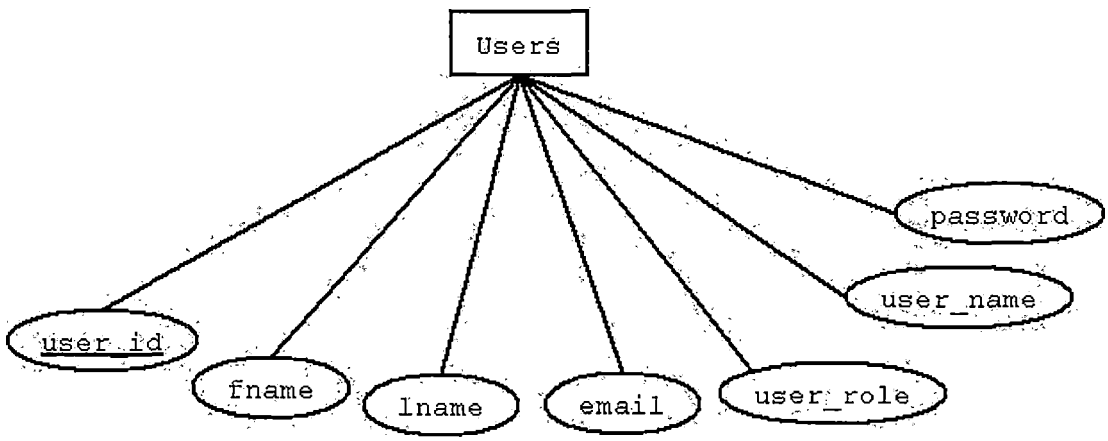

Figure 14. Entity Users

The characteristics of the attributes of the entity users are explained below in the table:

Table 2. Attributes of Table Users

Sno.	Field Name	Description
1.	User_id	Primary key, auto-incremented number
2.	Fname	First name of the user
3.	Lname	Last name of the user
4.	Email	Email address of the user
5.	Phone	Phone number of the user
6.	User_role	The role, Systems Administrator, Student or Faculty
7.	User_name	Unique username of the user
8.	Password	Unique Password of the user

2. Table proposal: This stores the information of the project proposal of the student. This has relationships with the table student-proposal which maintains the fact that there could be only one proposal for each student. An exception will occur if this is not maintained.

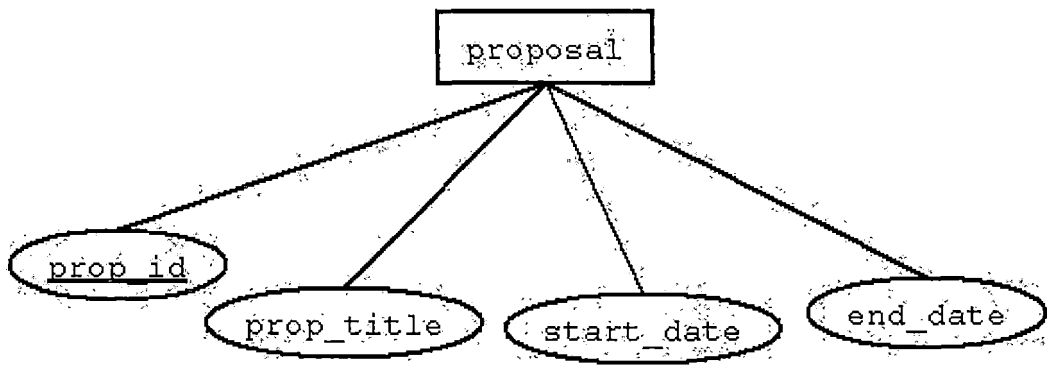

Figure 15. Entity Proposal

The characteristics of the attributes of the entity proposals are:

Table 3. Attributes of Table Proposal

Sno.	Field Name	Description
1.	Prop_id	Primary key, auto-incremented number
2.	Prop_title	Title of the Project Proposal
3.	Start-date	Date the proposal creation started
4.	End_date	Date the proposal was approved

3. Table student_proposal: This stores the attributes that link the users with the proposal table. The

attributes of this table make sure that there is only one proposal per student.

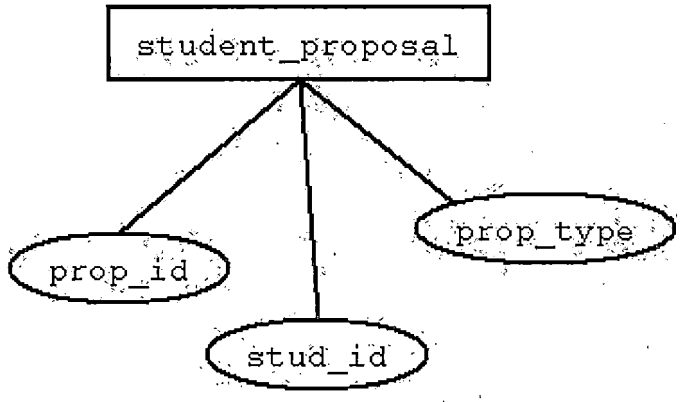

Figure 16. Entity Student_Proposal

The characteristics of the attributes of the entity student_proposal are:

Table 4. Attributes of Table Student_Proposal

Sno.	Field Name	Description
1.	Prop_id	Auto-incremented number, id for proposal
2.	Stud_id	Refers to the user_id of the users table
3.	Prop_type	Type of proposal, Strict IEEE or user-defined.

4. Table committee_members: This table stores the information of who the committee_members are as well as their status in the committee and review.

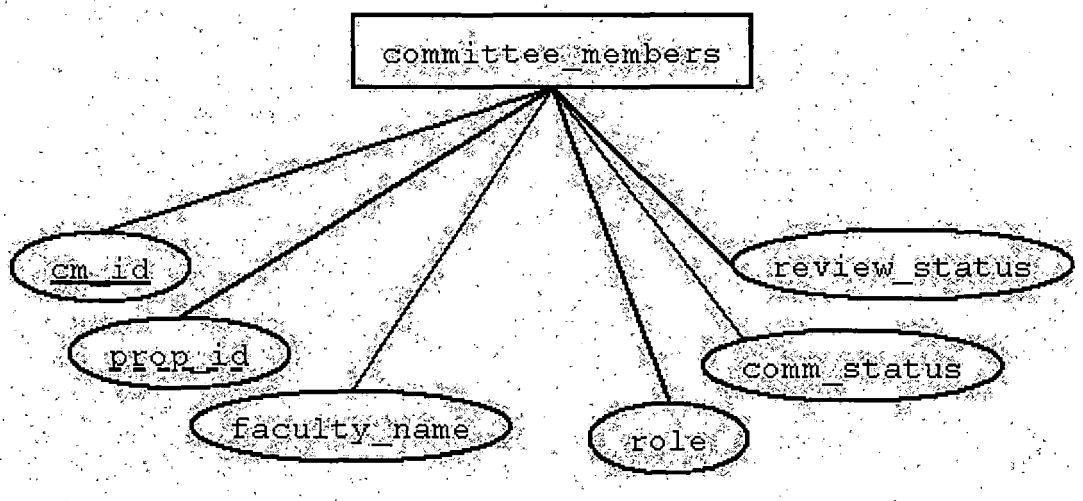

Figure 17. Entity Committee_Members

The characteristics of the attributes of the entity committee_members are explained in detail in the table shown below. These are linked with two table attributes namely proposal and users.

Table 5. Attributes of Table Committee_Members

Sno.	Field Name	Description
1.	Cm_id	Auto-incremented number, id for committee_member
2.	Prop_id	Refers to the prop_id of the proposal table
3.	Faculty_name	Last name of the faculty member
4.	Role	The role, advisor or member of the faculty
5.	Comm_status	The comm.. status. If the member has been approved by the advisor or not approved.
6.	Review_status	The proposal review status, reviewed and approved, reviewed and commented or commented and approved.

5. Table sections: This table has all the data related to the section, including the text and the images. This table is created individually for each proposal.

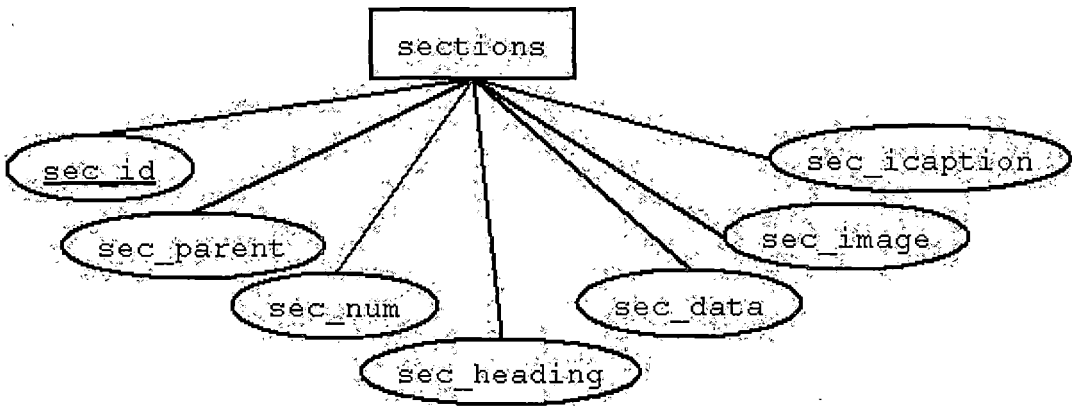

Figure 18. Entity Sections

The characteristics of the attributes of the entity sections are:

Table 6. Attributes of Table Sections

Sno.	Field Name	Description
1.	Sec_id	Auto-incremented number, id for section
2.	Sec_parent	Section id of the parent of the section
3.	Sec_num	Number of the section based on its position in the table of contents
4.	Sec_heading	This is the heading of the section. This is un-editable in Strict IEEE based proposal but editable in user-defined proposal
5.	Sec_data	Section text.
6.	Sec_image	Path of the image stored here
7.	Sec_icaption	Caption of the image

6. Table comments: This table stores the comments made by the committee members based on the sections and replies to those comments by the student. This table is created individually for each proposal.

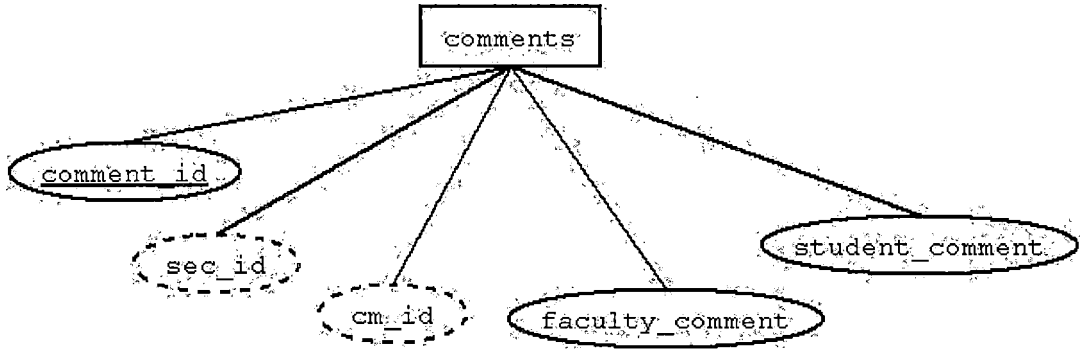

Figure 19. Entity Comments

The characteristics of the attributes of the entity comments are:

Table 7. Attributes of Table Comments

Sno.	Field Name	Description
1.	Comment_id	Auto-incremented number, id for comments
2.	Sec_id	Refers to a sec_id from table sections
3.	Cm_id	Refers to a cm_id from table committee_members
4.	Faculty_comment	Comment made by the faculty
5.	Student_comment	Comment made by the student

All the above mentioned entities are related and can be shown in an Entity Relationship diagram below. It also shows the cardinality of the entities joined in a relationship.

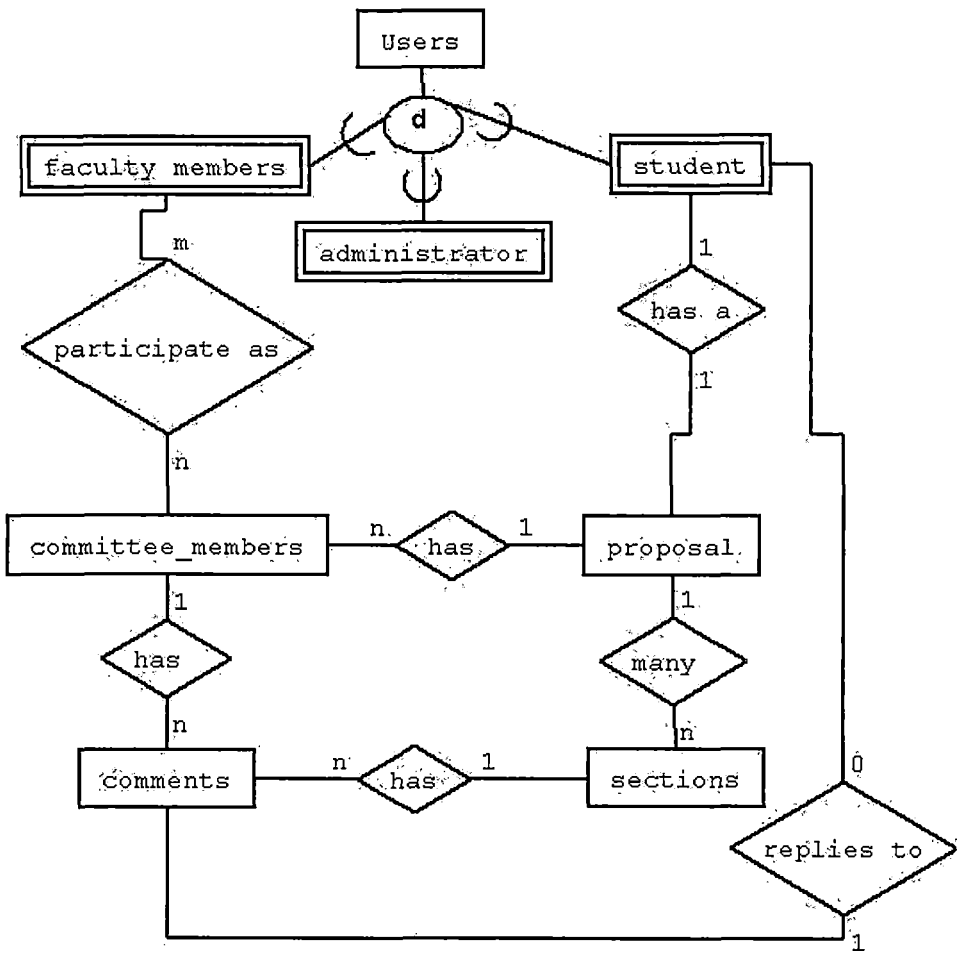

Figure 20. Athena's Entity Relationship Diagram

2.4 System Setup

Athena's development system can be installed and constructed from the starting by installing and running the following applications or software's on the actual server on which Athena will reside for users to access.

Installation of Java Development Kit 1.5.01

The Java 2 SDK is a development environment for building applications, applets, and components using the Java programming language. The Java 2 SDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform. These tools are designed to be used from the command line. Except for the appletviewer, these tools do not provide a graphical user interface. Do not install any of the older versions than this as they will not have all the packages required for Athena.

Install Tomcat - Servlet/Java Server Pages Container

Athena uses Tomcat version 5.0.30 which implements the Servlet 2.4 and JavaServer Pages 2.0 specifications from the Java Community Process, and includes many additional features that make it a useful platform for developing and deploying web applications and web services.

Install MySql Database Server

Athena uses MySql version 4.0.23-nt. The MySql (R) software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySql Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software.

Install Java Database Connectivity Driver

Athena uses MySql Connector/J which is a native Java driver that converts JDBC (Java Database Connectivity) calls into the network protocol used by the MySql database. It lets developers working with the Java programming language easily build programs and applets that interact with MySql and connect all corporate data, even in a heterogeneous environment. MySql Connector/J is a Type IV JDBC driver and has a complete JDBC feature set that supports the capabilities of MySql. Make changes in the server.xml and web.xml files in the application server to establish connectivity between tomcat and MySql.

Test Web Application

All these software's are available on the Internet for free. The set up can be done on any operating system for the server side. Athena was developed on a Windows XP

Home Edition. But will be served from a Linux system when ready for deployment.

The client can have any operating system such as Windows 95/98/NT/2000/XP workstation, Mac OS, OS/2, UNIX, Linux etc. The browsers that the client can use for viewing HTML documents and using Athena are Netscape Navigator 3.0 or higher or Microsoft's Internet Explorer 3.x or higher.

2.5 Summary

The software design of the project was presented in Chapter Two and it underlines the structure of the application developed. Athena was a 3-tier architecture based web application that provides interfaces in HTML for the clients. The data is retrieved from a MySql database through JavaBeans and sent to the browsers by JSP pages.

CHAPTER THREE

SOFTWARE QUALITY ASSURANCE

3.1 Introduction

This chapter provides the procedures in which Athena - Online Proposal Development System was tested and the results gathered from it. It was tested on Microsoft's Internet Explorer Browser.

3.2 Unit Test Plan

Athena - An Online Proposal Development System was a web based application that helps Masters Student's create their project proposals online and get them reviewed by their committee members, irrespective of which geographic location they are at.

As part of the Unit Test Plan, Athena was tested based on the user types since each user has its own set of interfaces. All the interfaces are web based and available through the browser.

As part of the unit test plan these steps were taken and results found:

1. All the hyperlinks available in each set of interfaces of Athena were checked, irrespective of the content that they provide. It was found that all of them do work.

2. User data input was given to check if the presentation logic worked. The presentation logic was implemented in JavaScript which check if the input taken from the users was valid. The display was in HTML. The Interfaces of Athena that required user input successfully validated it.
3. There are error pages that display the errors based on the functionality.
4. The errors in the input of entries are displayed in alert boxes by JavaScript.
5. The basic rules of web based applications of readability and presentation for each set of Interfaces was verified. The data and content was readable and displayed in a user friendly manner.

3.3 Integration Test Plan

As part of the Integration Test Plan, Athena was tested for functionality based on its user-type. The users have their own interfaces and after being tested as single units and their presentation they were tested based on functionality.

The functions of each user are as follows:

A. User: Systems Administrator:

- Create user account
- Edit user account
- Delete user account
- View User Information

B. User: Student

- Manage Project Information
- Manage Committee Information
- Choose Proposal Type
- Create Table of Contents
- Manage Proposal Text
- Manage Proposal Images
- View Proposal
- Check Status of Review
- Review Comments
- Contact Committee Members

C. User: Faculty Member

- Manage Committees
- Review Proposals
- Contact Students

Every set of functions are available on the left hand side of the browser as a menu.

The following tests were performed and results obtained:

1. Athena's user interfaces were tested now for content across each page depending on the functionality.
2. Every function of each user was tested.
3. A user was created as part of the administrator pages. After the information was entered the functions of edition, deleting and final viewing of all information was done. Athena's Systems Administrator pages tests passed successfully.
4. The student functionality was tested for bugs and limitations. Each function was checked for content and right response.
5. Certain limitations were found which have been documented and the bugs were fixed.
6. The faculty member pages were checked for functionality by itself as well.
7. There was some functionality of the student and faculty members that were interlinked and will be checked during system test.

3.4 System Test Plan

As part of the System Test Plan, the following tests were performed:

1. All those functionalities that interlinked and depended upon two different kinds of users were checked. This involved the very first fact that students and faculty members have to be authorized to access the system and they cannot use Athena until the Systems administrator entered their information in the database. Tested unauthorized users cannot view pages. They cannot get past the login page.
2. Unless a student creates a proposal the faculty will not be able to annotate on it. Tested, the list of proposals did not have the name of the students whose proposals have not yet been added to the database.
3. Unless the advisor approves the committee members they cannot see the proposal in their lists and therefore cannot review or annotate on them.
4. Unless the faculty annotates on the proposal, the student will not be able to view them. Tested and was successful.

5. Once the annotation is done, the student can view them and make changes to the proposal. This was checked to see if it works and it did.
6. It was checked if the system could handle more than one faculty member accessing the same proposal. It was found that they can access it, as they are read-only versions of it. The student is the only user who has the right to edit the proposal.
7. All functionality errors and exceptions were displayed in error pages and exception pages that give a user-friendly error message to the user to let them know that they have either entered a wrong input, or something that does not go with the functionality of the system.

3.5 Summary

Testing is an important part of application development. And after testing Athena its limitations were made clear and all the bugs were tried to be fixed or the functionality added to the limitations of this prototype. Each testing phase brought Athena closer to its goal of being user-friendly, easy to understand system for the Department of Computer Science.

CHAPTER FOUR

MAINTENANCE

4.1 Introduction

This chapter provides the measures that need to be taken to maintain Athena - An Online Proposal Development System. The maintenance that needs to be done if:

1. Issues arise on the front-end of the application or the back-end programming.
2. Issues with the tools utilized for development.
3. Issues with the directory structure of the application.

4.2 Maintenance Guidelines

These guidelines aim at providing the information for maintenance issues that might arise depending on the user-type and the front end of the application as well as depending on the programming and servers at the backend of the application.

Front-end Interface Management

The front-end interface creation was done in HTML. A webmaster will be needed to maintain the system and except for basic HTML no other professional skills are needed.

Application Interfaces Management

The programming has been done in JSP and Java and Athena has been tested for all the functionality. But if issues arise then the webmaster will have to know Java, JSP, JavaScript and MySql to troubleshoot problems in the business logic of the application. The Java API helps in the programming part of the JavaBeans and help can be found on the Internet at: http://java.sun.com/j2se/1.3/docs/api/ or the references [8] and [9]. The MySql help could be found on the Internet as well at http://www.MySql.com/doc/.

Administration

A Systems Administrator/webmaster is needed to frequently add new users and provide general maintenance of the system.

4.3 Tools Utilized

Athena was developed using the following tools:

1. Tomcat Web Server
2. MySql Database Server
3. Java 2 Standard Development Kit
4. JDBC Driver
5. Macromedia Dreamweaver
6. Dia

4.4 Directory Structure

Athena's used Apache's Jakarta Tomcat server as the JSP/Servlet container. The web application was developed in the "webapps" sub-directory of the Jakarta Tomcat main directory. It was stored in a directory under "webapps" called "mpds".

The files present at each directory level:

mpds directory

All the .jsp and .html files are saved here. The naming convention followed can be explained by an example of a create user admin page:

1. Presentation page: admin-ca.jsp.
2. JavaScript Code for user-input validation: admin-cachk.jsp.
3. JSP servlet page for communication with the JavaBeans that insert the data: admin-cainsert.jsp.

The common files used by all the users are placed directly under "mpds", but all the other pages are placed in sub-directories based on their user-type first and then their functionality. The sub-directories under "mpds" are:

1. admin - The administrator pages are saved here.
2. faculty - The Faculty pages are saved here.

3. student - The Student pages are saved here.
4. web-inf - This directory has the .java files stored in it and the respective .class files are stored in another sub-directory in it called mpds_modules.

MySql Directory

Athena's database is in the C:\MySql\bin directory. It needs to be accessed either from the WinMySqladmin console or from the command prompt. A user name and password are needed to access the database which is called "mpds".

In case of situations where issues are unsolvable or questions with the logic or programming practices are involved, the author and developer of Athena, Humaira Rahim can be contacted at humu888@yahoo.com.

4.5 Summary

This chapter will enable future users and developers of the next prototypes of Athena to understand its working technique, development mechanisms and procedures. This will facilitate in any new technologies that future users, developers, and administrators might find needful for the system.

CHAPTER FIVE

USERS MANUAL

5.1 Introduction

This chapter provides a detailed description about all the interfaces and functionalities of Athena.

Athena - An Online Proposal Development System was developed for composing, reviewing and annotating project proposals online. After the initial user authentication Athena divides its interfaces based on the type of user. Athena has three types of users:

A. Systems Administrator

B. Student

C. Faculty Member

The following sections give a detail of the web interfaces of Athena.

5.2 General Pages

There are two general pages, one the welcome page for Athena and the other is the Login page. Each time a user logs off it is taken to this page.

This is the System Athena's index or default page which means this is the page that loads when it is accessed from the browsers of the users. This page directs the user to the Login page.

Welcome Page

Athena has a welcome page that gives a brief description about its functionality and welcomes the users to its workspace. It has a link that directs authorized users to the login page.

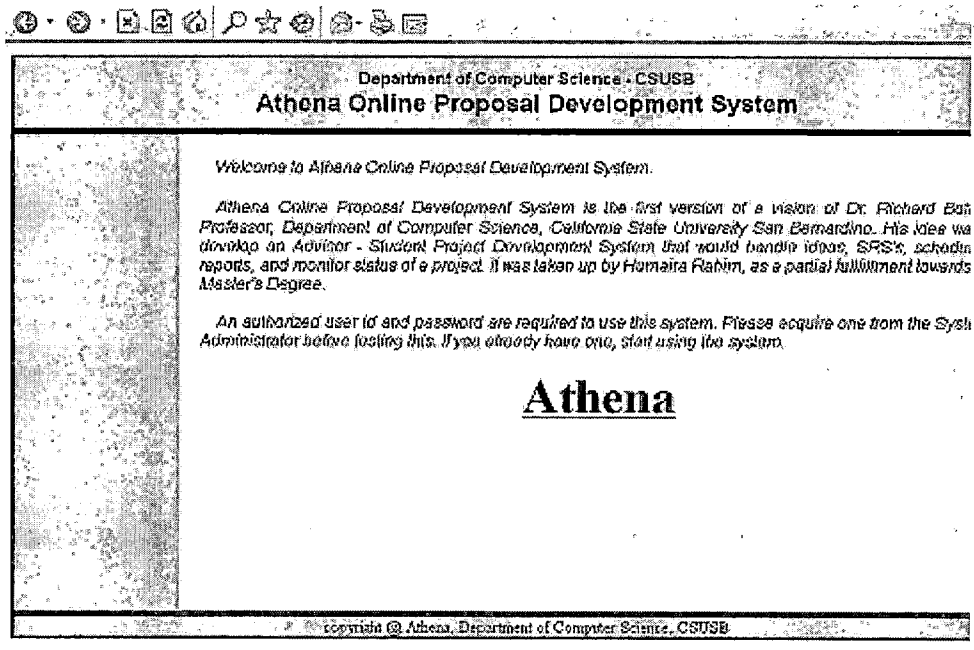

Figure 21. Athena Welcome Page

Login Page

This page authenticates users based on their username, user-type and password. This sends a request to the JavaBean that establishes connection with the database and checks information. On approval, it forwards it to the set of Interfaces assigned to that user-type.

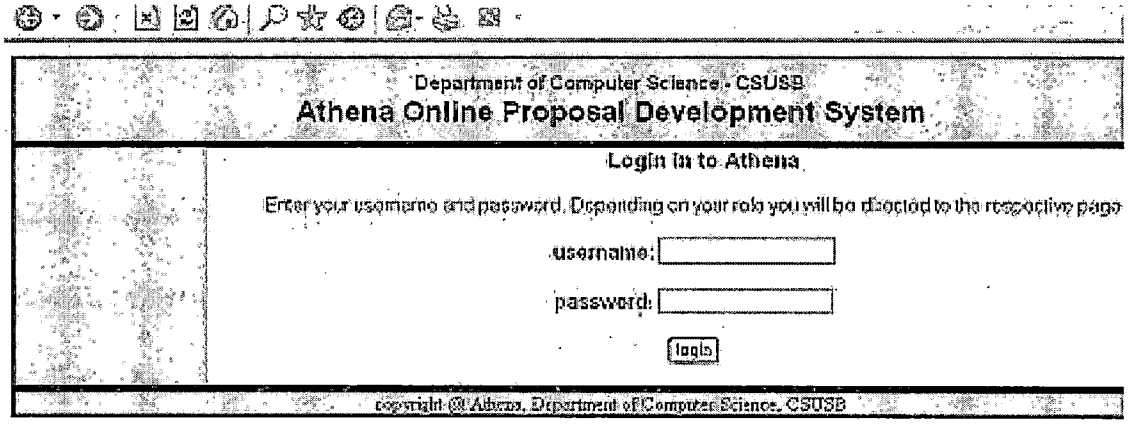

Figure 22. Athena Login Page

5.3 Systems Administrator Pages

These pages are used by the administrator to maintain the information of the authorized users of the system, they are as follows:

Administrator Main Page

This is the main page of Administrator.

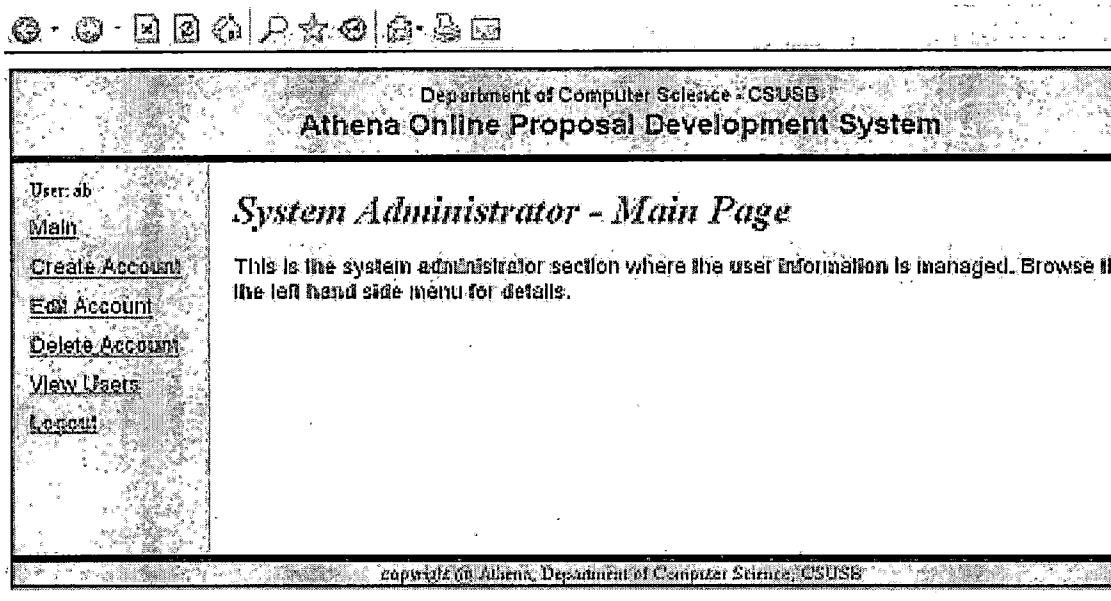

Figure 23. Administrator Main Page

Create User Account Page

The administrator enters the user information and makes an account for the user.

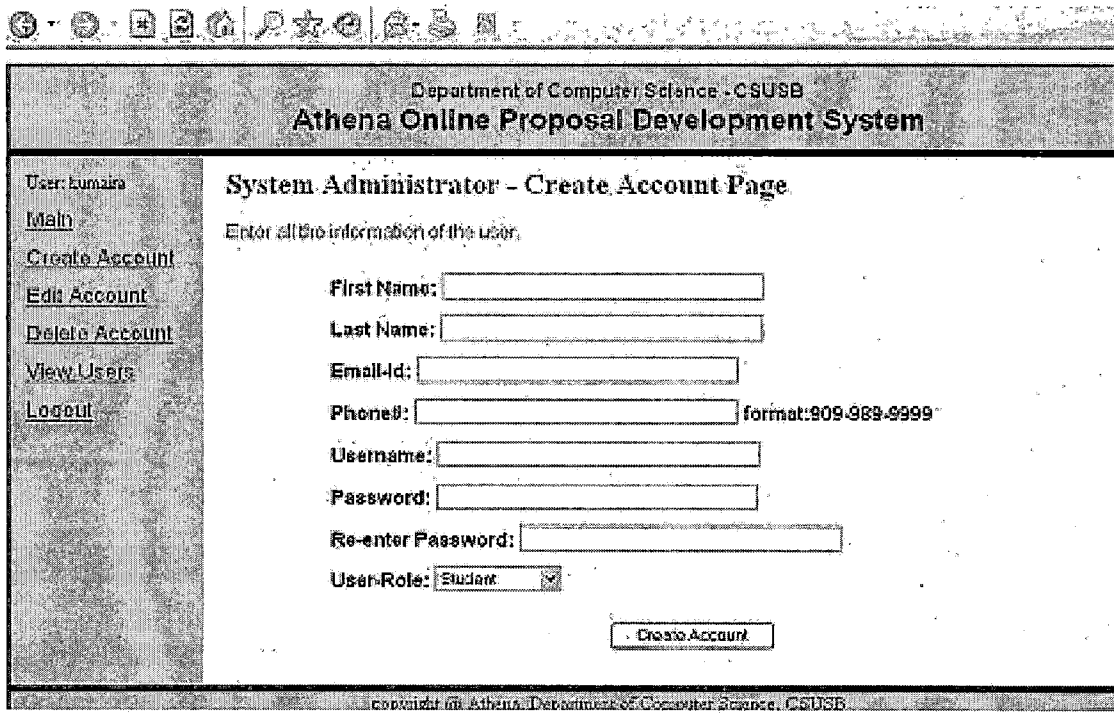

Figure 24. Administrator - Create User Page

Edit User Account Page

The Administrator can edit the user information at the request of the user.

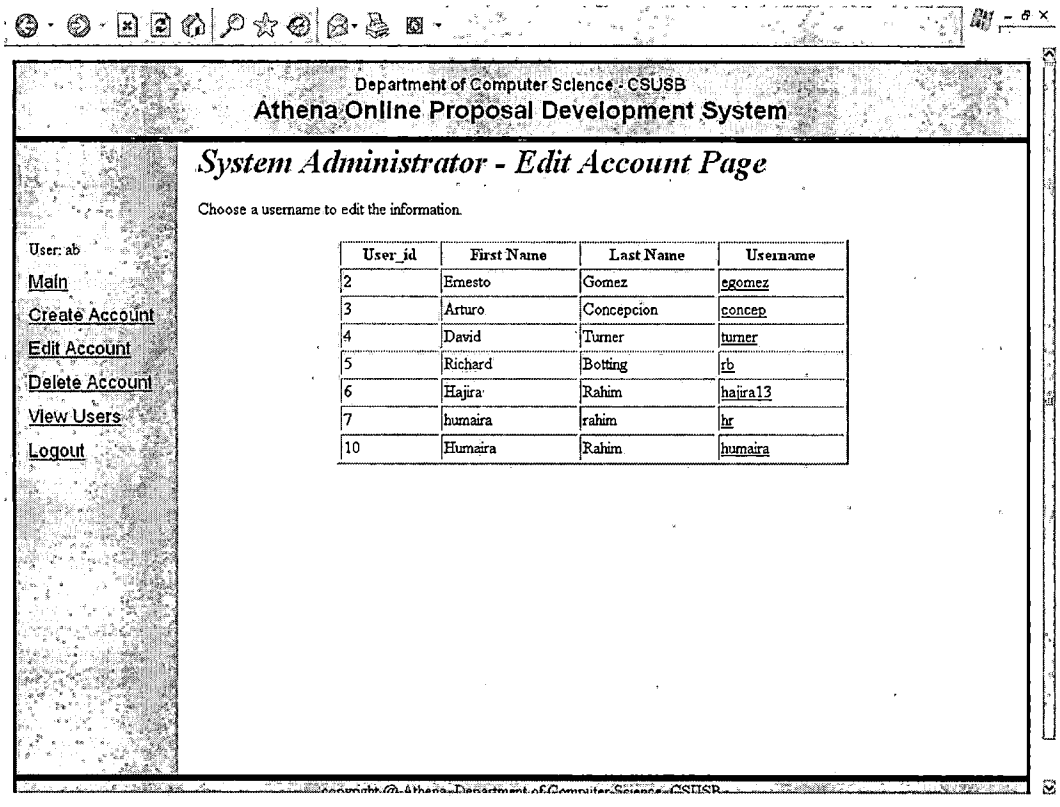

Figure 25. Administrator - Edit User Page

Delete User Account Page

Once the student advances to candidacy they are deleted from the system. If any faculty member leaves the department then the administrator deletes the information stored for that particular faculty.

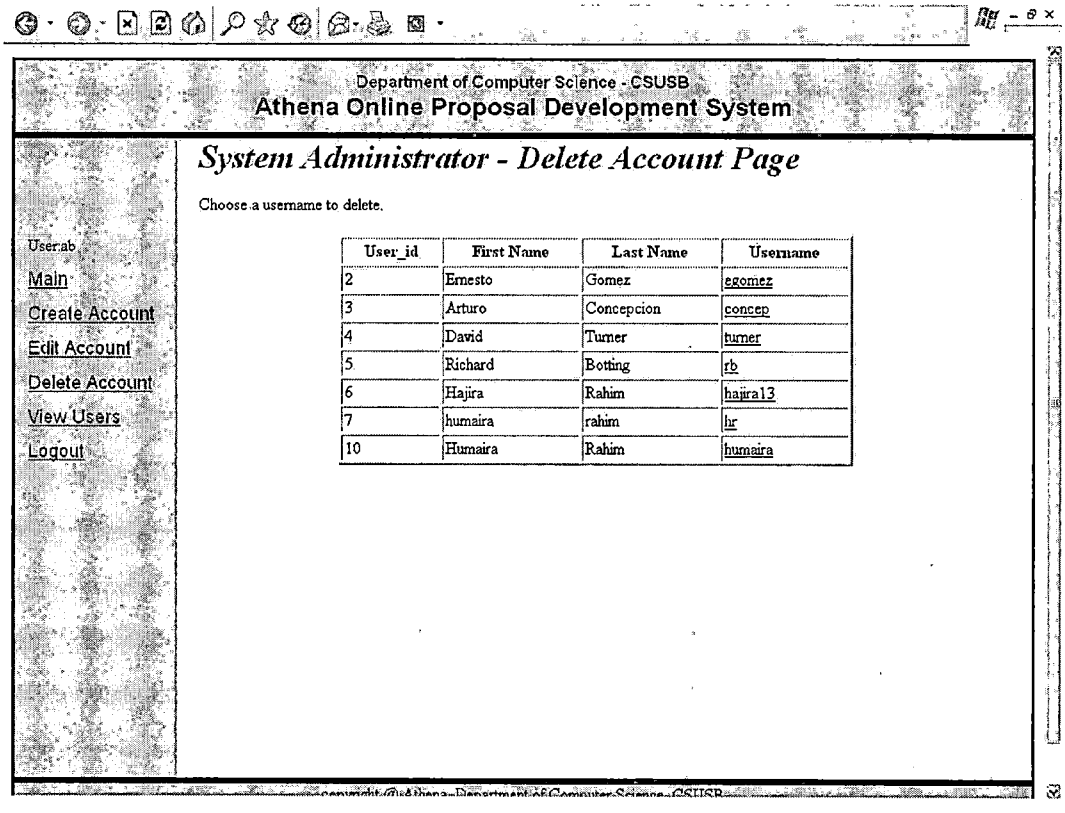

Figure 26. Administrator - Delete User Page

View All Users Page

All the information of the all the users of the system can be viewed from here.

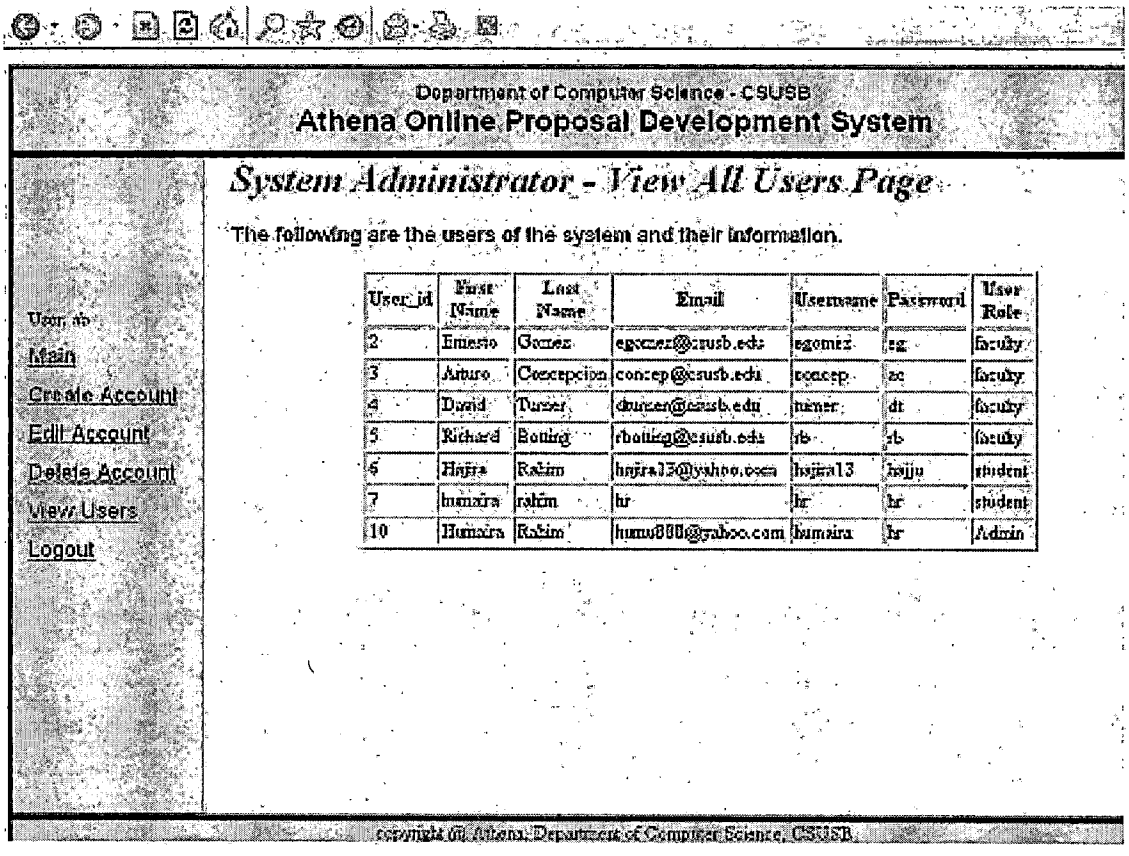

Figure 27. Administrator - View All Users Page

5.4 Student Pages

This set of pages is used by the students to compose, modify and get approval of their proposals. They are:

Student Main Page

This is the student main page that describes in brief the functionality of the user.

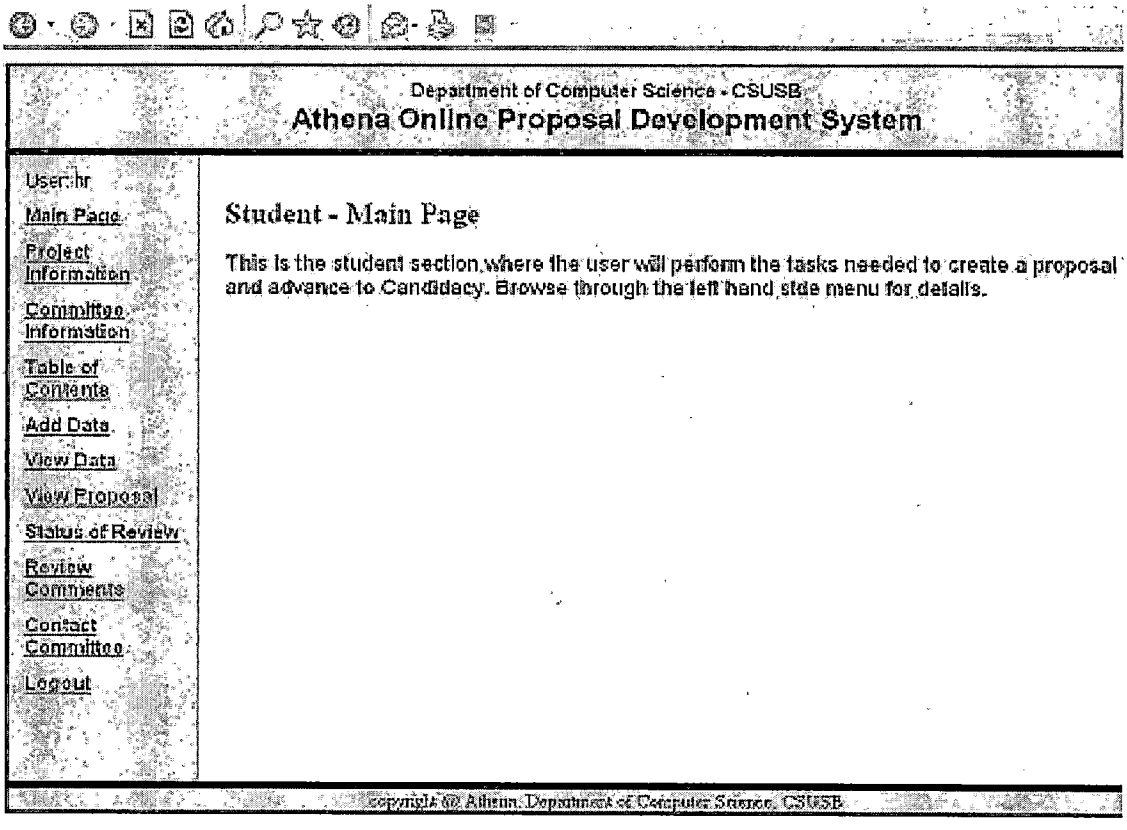

Figure 28. Student - Main Page

Project Information Page

The student will enter the project information in this page and the next time he/she will login they can see the information they had entered. They can change information of they want to.

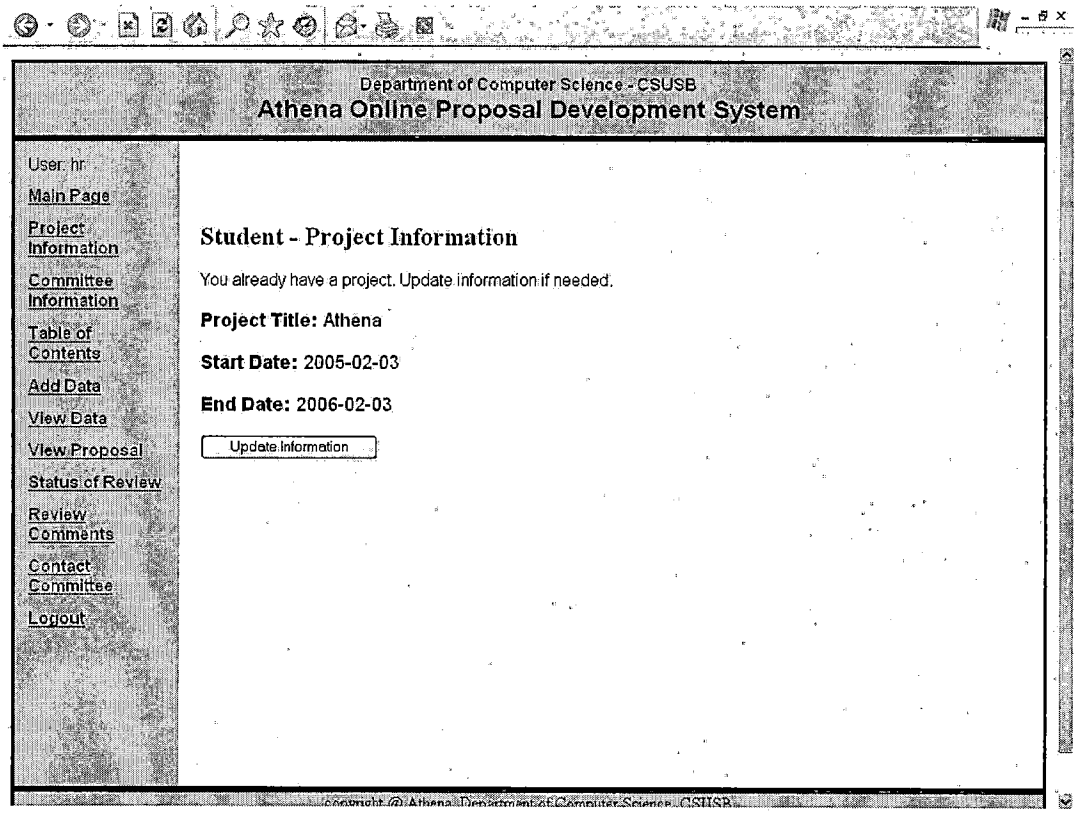

Figure 29. Student - Project Information Page

Committee Information Page

The student enters the committee members for the project after getting approval from them. Each of those faculty members will see the listing of this students proposal when they log in. They can contact that student if they do not wish to be on their committee or if there has been a mistake.

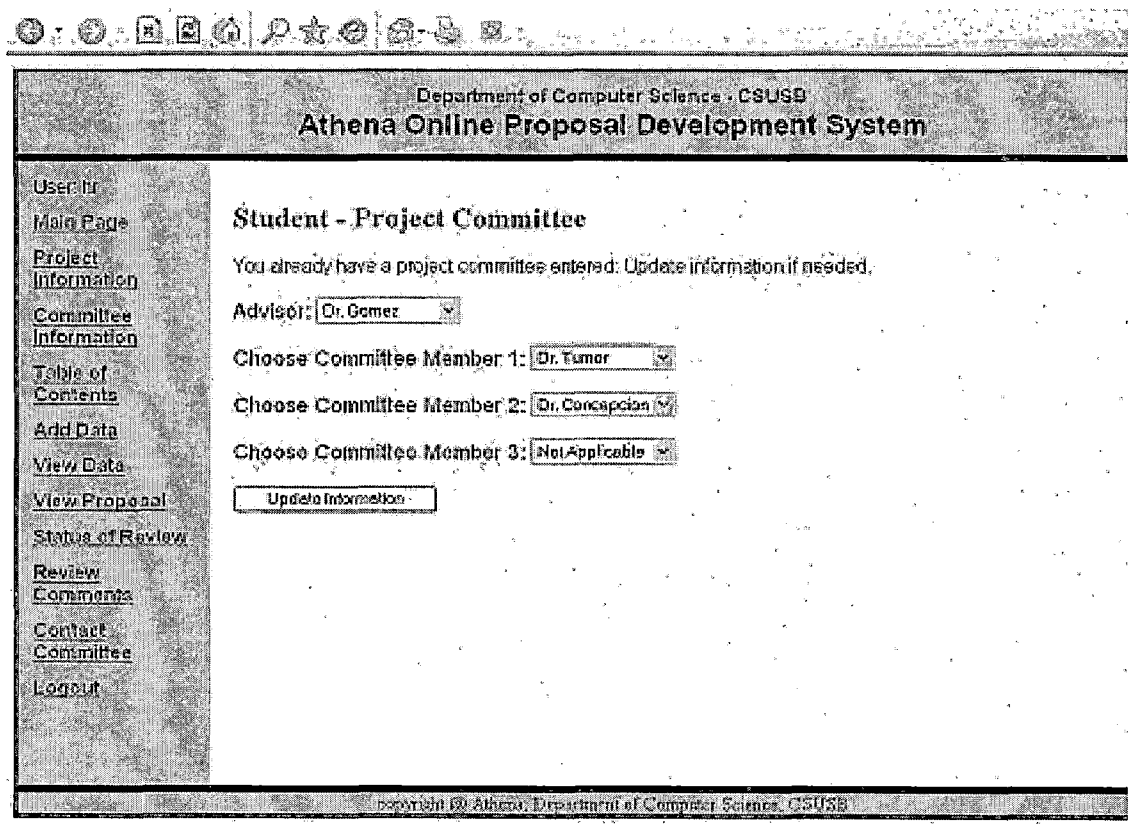

Figure 30. Student - Committee Information Page

Choose Proposal Type Page

The student chooses the type of Proposal for the project, either Strict IEEE based or IEEE based user-defined. IEEE uses the IEEE std. 830-1998, IEEE Recommended Practice for Software Requirement Specification - Annex A [5].

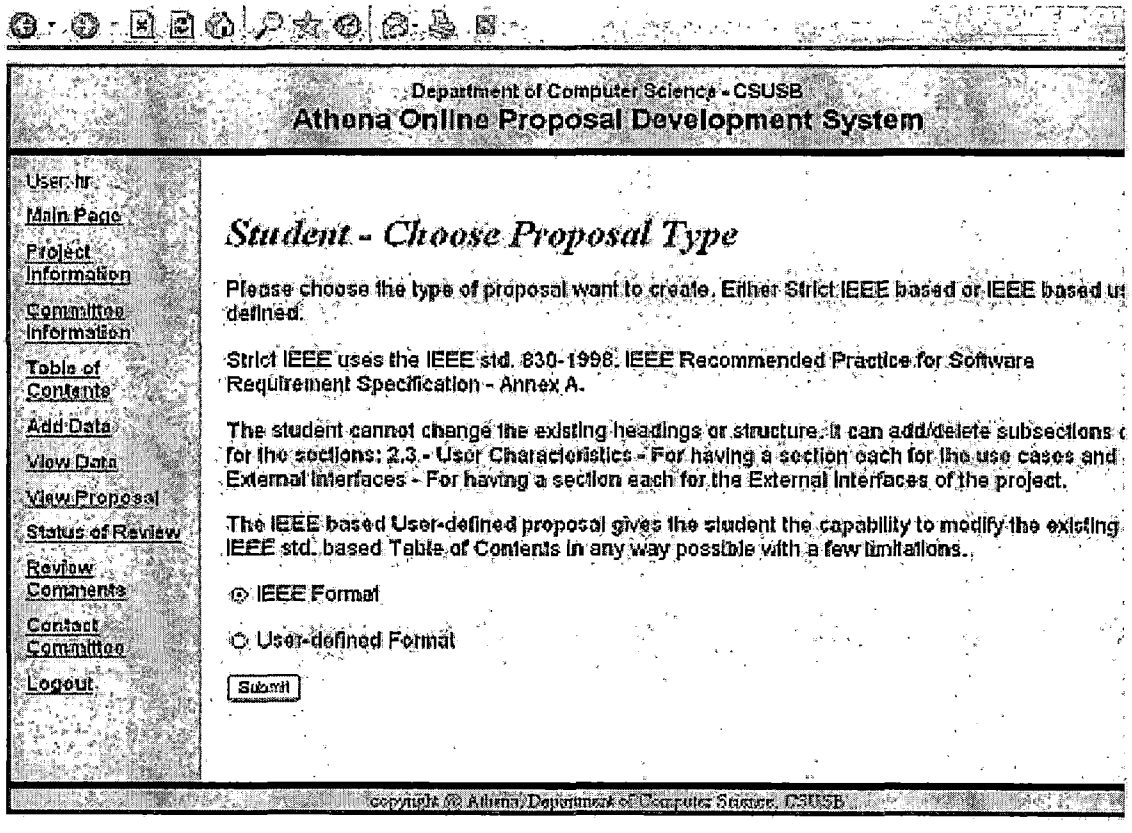

Figure 31. Student - Choose Proposal Type Page

Standard Based Proposal Page

This is the table of contents page that has been decided upon based strictly on the IEEE standards [5].

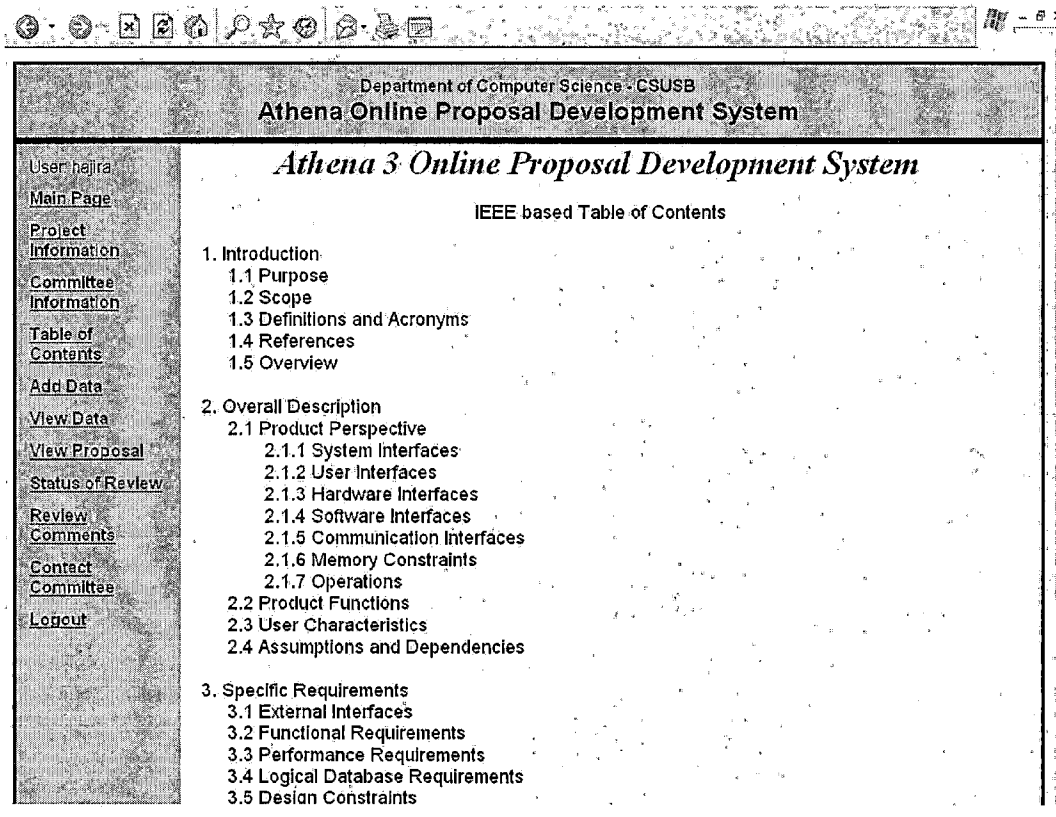

Figure 32. Student - Standard based Proposal Page

User-Defined Proposal Page

This is the table of contents page that has been decided upon based on a survey and also the IEEE standards [5].

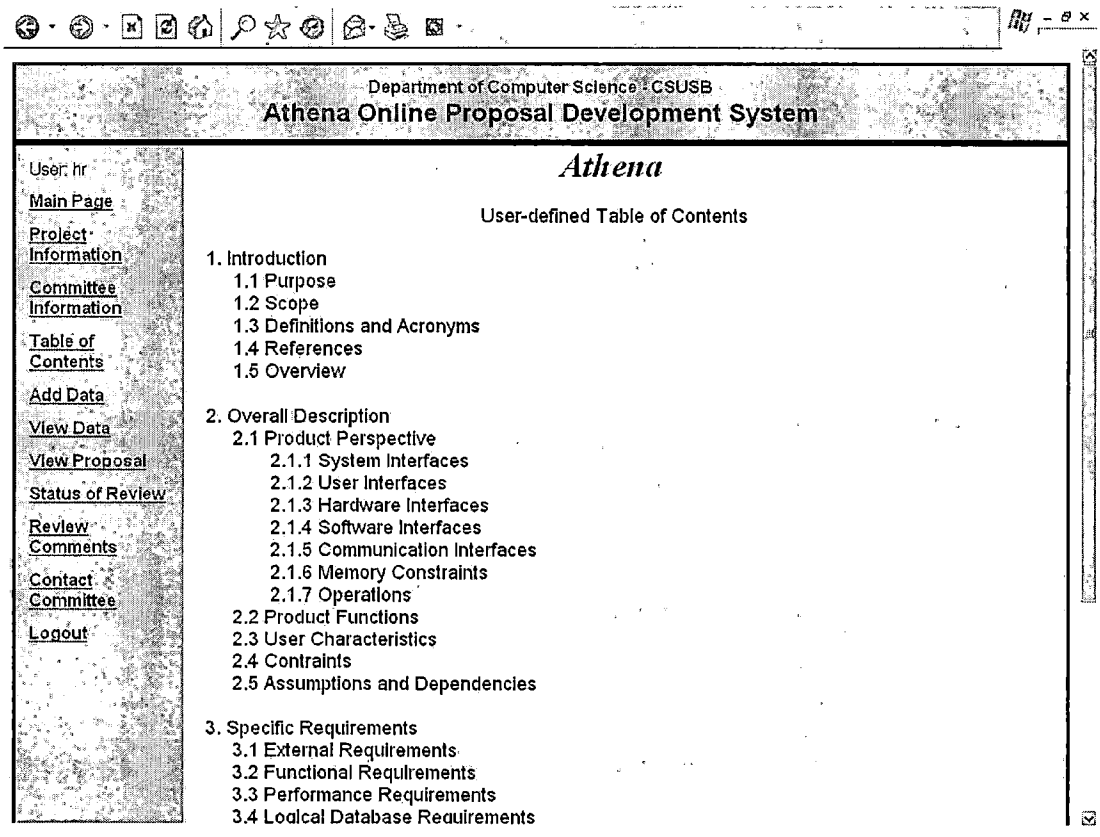

Figure 33. Student - User-Defined Proposal Page

Edit Table of Contents Page

The headings of the sections can be edited as well as insertion and deletion of sections is taken place here. The student customizes the Table of contents to match its own. The 3 chapters cannot be deleted and no new chapters can be added. But sections and subsections can be inserted or deleted.

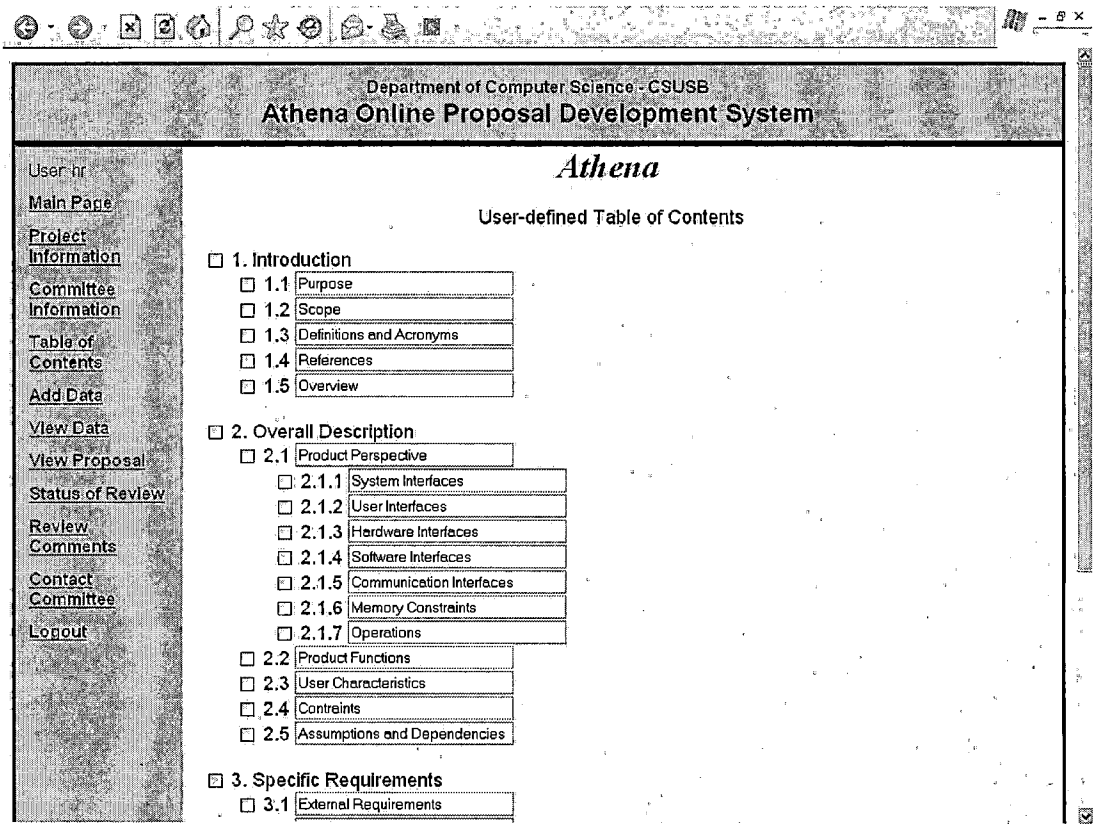

Figure 34. Student - Edit Table of Contents page

Add Data Page

On completion of customizing the table of contents the student then can go on and either write the data in based on the table of contents or copy it from a text editor in this page into the Athena database.

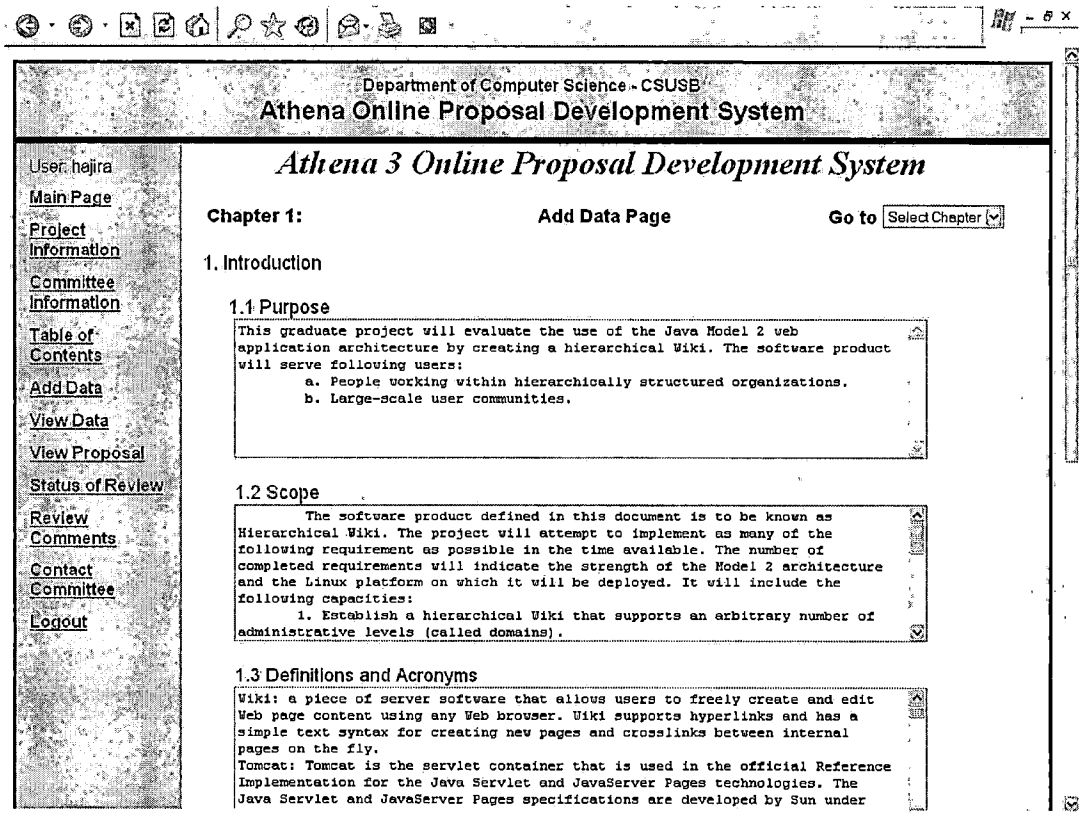

Figure 35. Student - Add Data Page

View Data and Add Image Page

The student can check the data entered by selecting a chapter in the proposal. The "add image" buttons can be seen with each section for the user to browse for images and add them in the proposal wherever needed. The limitations with the addition of images are:

a. There can only be one image per section.

b. The images can be of jpeg, gif, bitmap and png.

c. The maximum size of an image is 1 MB.

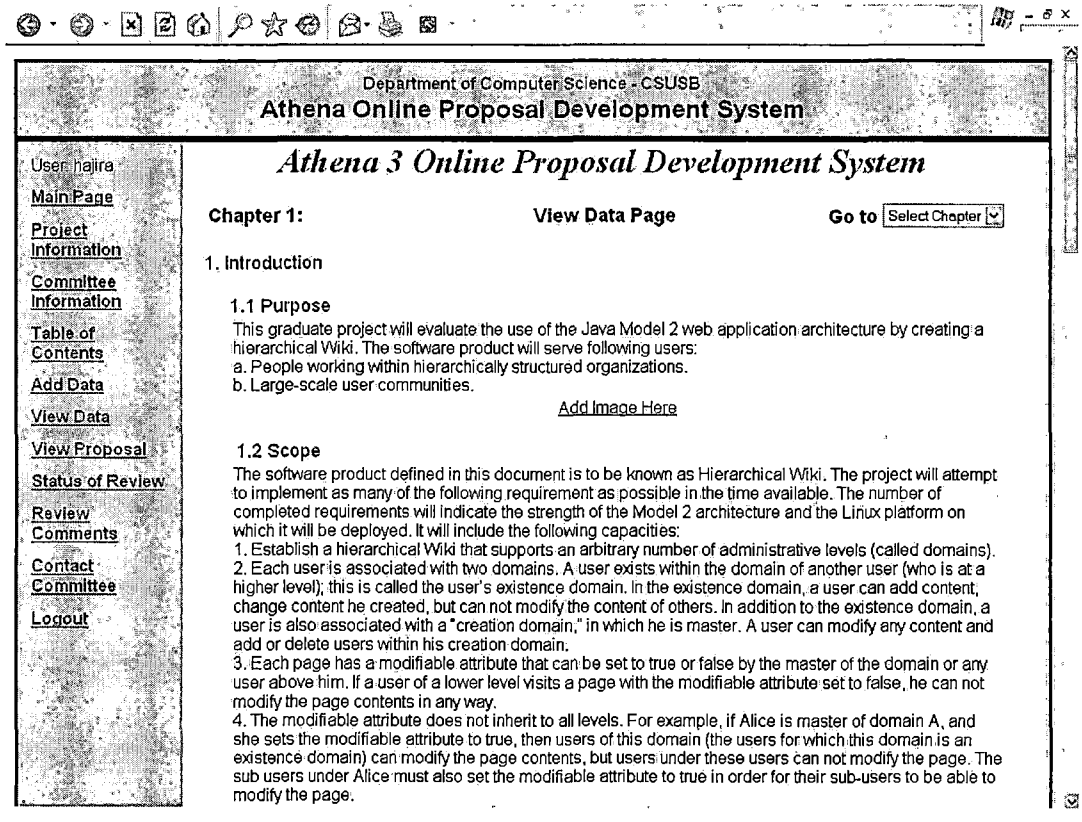

Figure 36. Student - View Data/Add Image Page

View Complete Proposal Page

Once the proposal is complete the student can also check the data in read only mode as it will appear to the committee members.

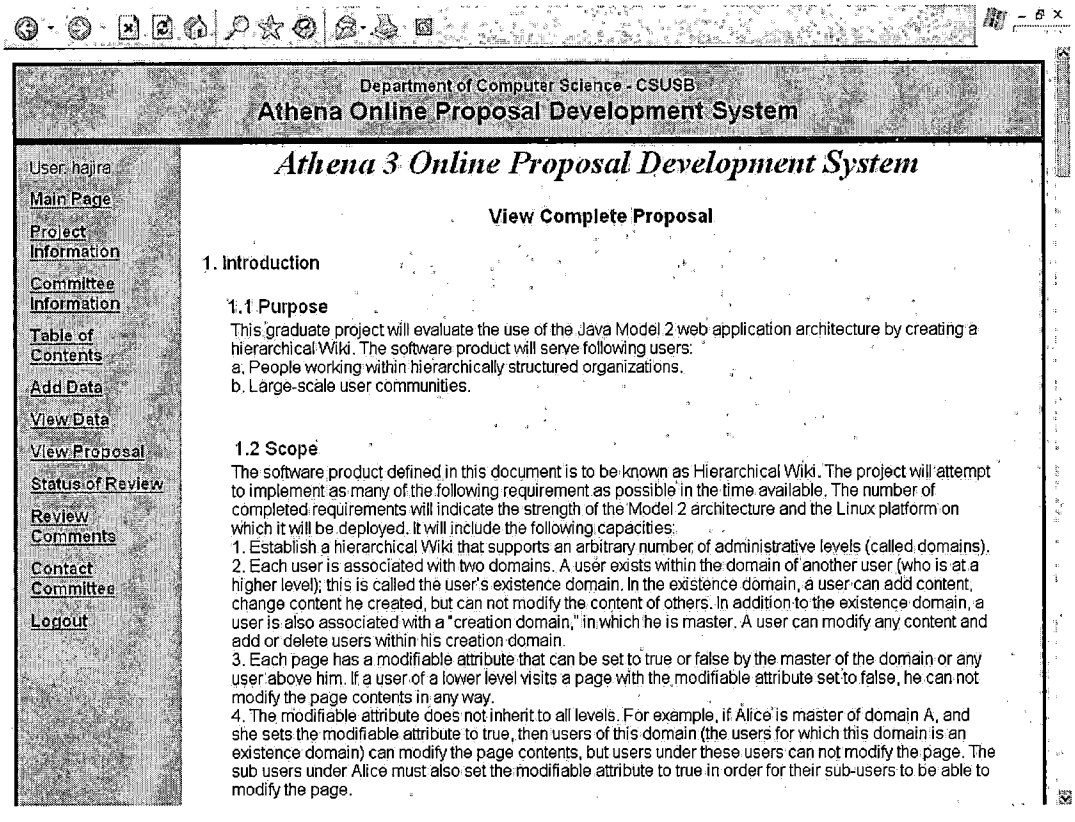

Figure 37. Student - View Complete Proposal Page

View Complete Proposal Page

Once the proposal is complete the student can view comments made by the committee and reply to them if they want to. This is done in the Student Review Comments Page.

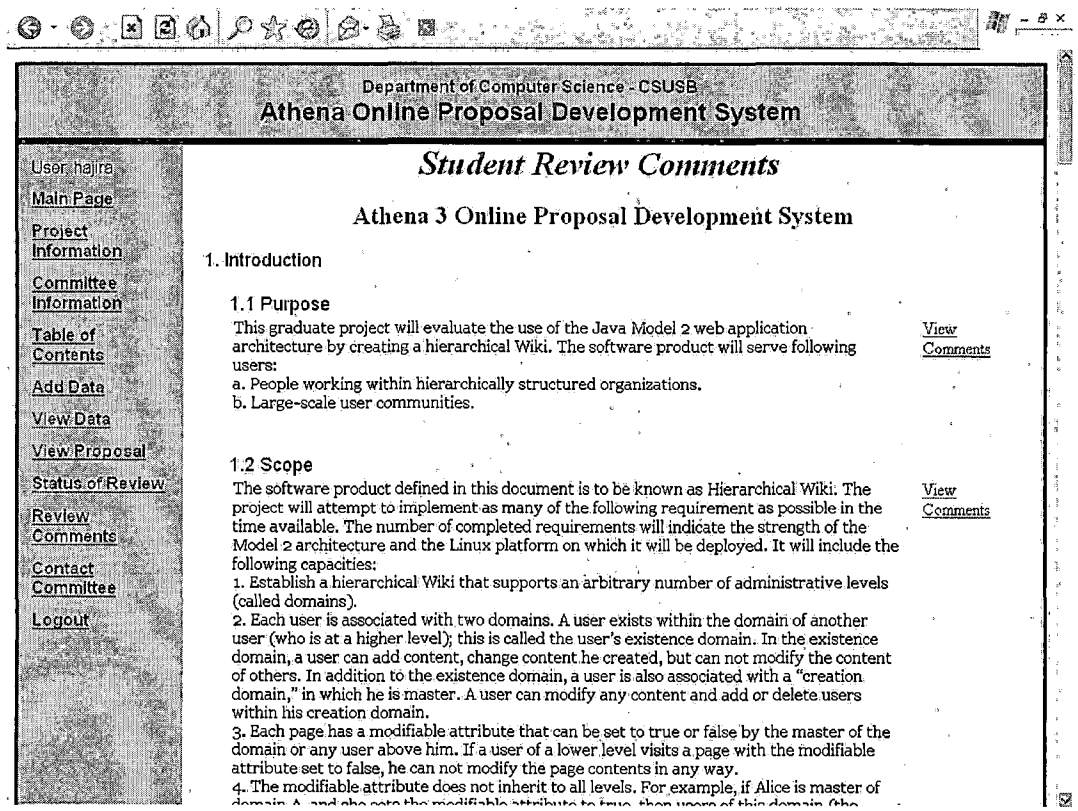

Figure 38. Student - Review Comments Page

Contact Committee Members Page

The Student can contact the committee members by email. The System gets the emails of the committee members automatically into the "To" field of the message.

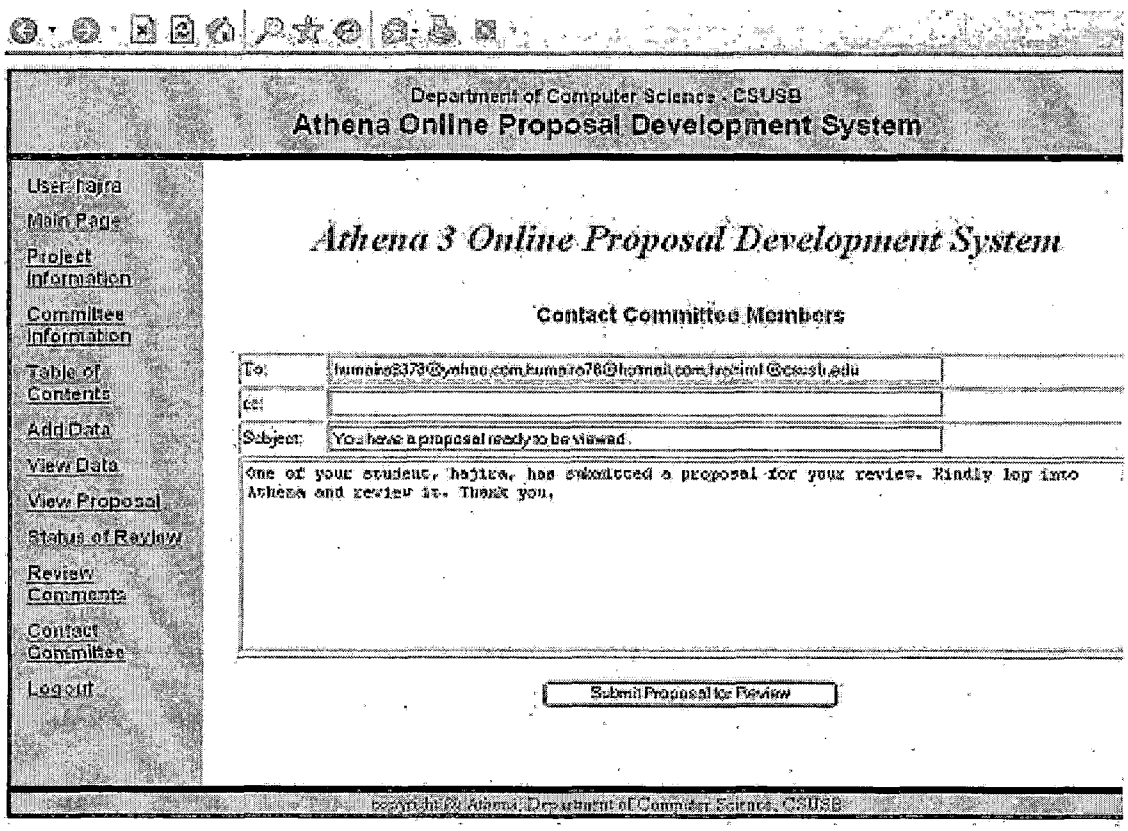

Figure 39. Student - Contact Committee Page

5.5 Faculty Member Pages

The Faculty Members play roles of either an advisor or a committee member in the review process of proposals. Their functions are shown in the following pages:

Manage Committees Page

This page displays all the projects that the faculty member is a part of. The page is divided into two sections where the role is as advisor and role is as committee member.

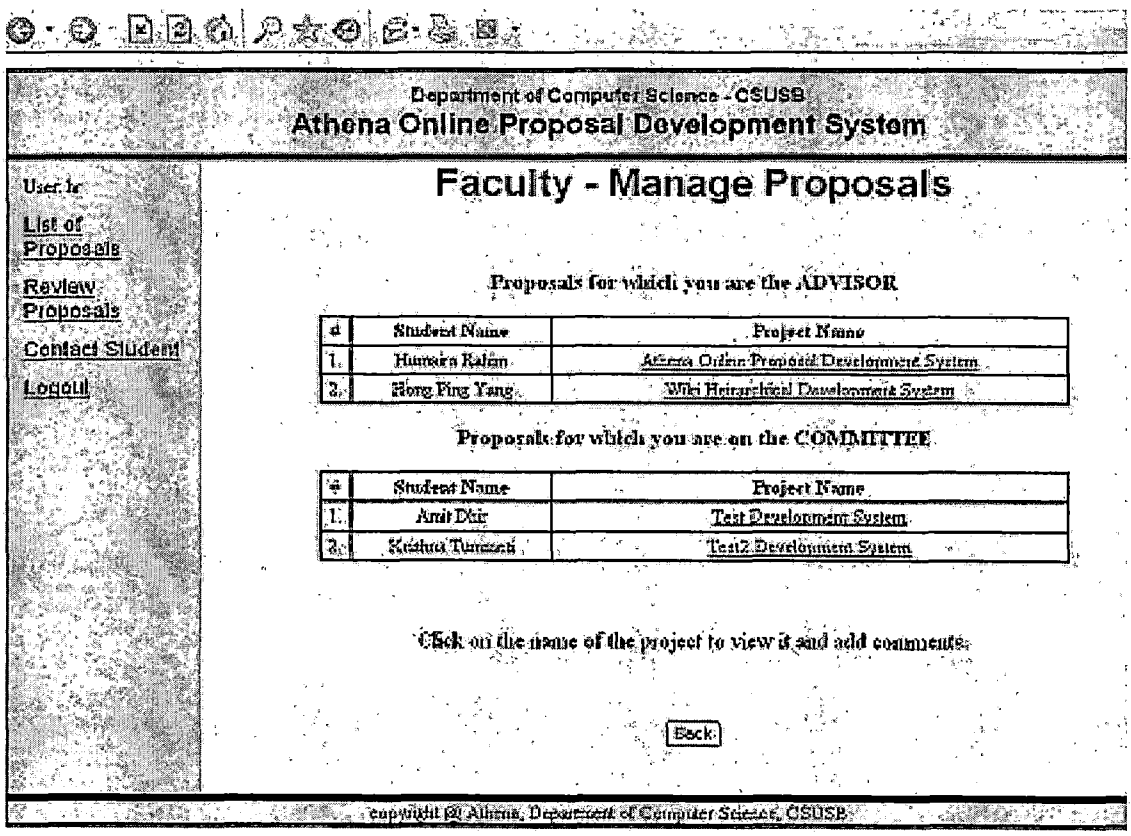

Figure 40. Faculty - Manage Committees Page

Review Proposals Page

This is the page on which the faculty members gives the comments. The comments are directly stored in the database. A pop-up window appears whenever the faculty member wants to comment on a section of the proposal. All comments are saved in the database and can be retrieved by the student or the other committee members.

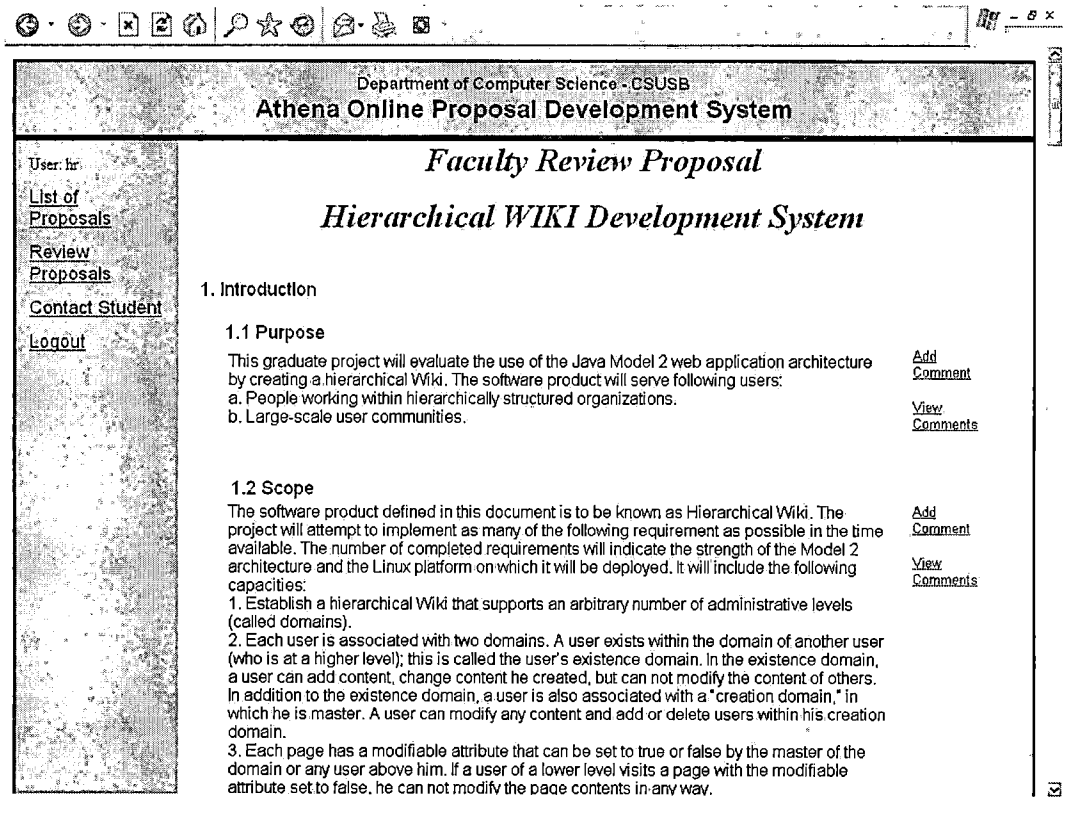

Figure 41. Faculty - Review Proposals Page

Contact Student Page

The faculty member can contact the student by email through the Athena workspace.

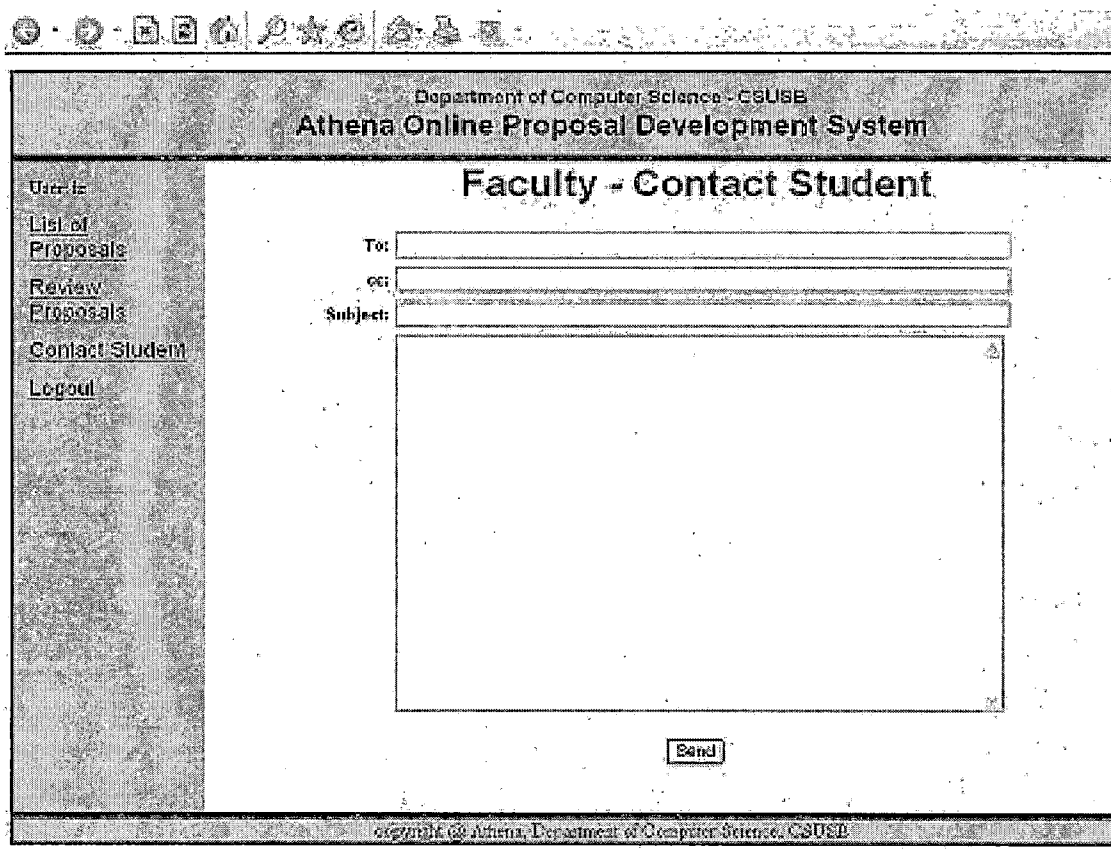

Figure 42. Faculty - Contact Student Page

The interfaces were designed to be readable and clear for all the users. The contrasting colors were used to facilitate for users with color blindness or difficulties. The font size has also been assigned a readable size.

5.6 Summary

This section explains in detail the user interfaces of the web application based on user-type and how they are interdependent and what functionality of the system each of them implements.

CHAPTER SIX

CONCLUSIONS AND FUTURE DIRECTIONS

6.1 Introduction

Athena - An Online Proposal Development System was a system developed to help Masters Students' and their committees in composing, reviewing and annotating proposals in an efficient, quick and reliable environment. It was developed using JSP, JavaBeans and MySql Query Language. The proposal data and project information is saved in the database and retrieved as needed.

6.2 Conclusions and Benefits

Proposal Development is an important feature of an educational system. The conclusions that could be drawn from the system are numerous. It has many benefits over the related systems like Zeus, Poseidon and Tales. Some of the conclusions and benefits are:

1. No matter where the student or committee members are geographically located, they can compose, review, annotate and communicate easily. All they need is an Internet connection.
2. There will be not be any repetition of comments as each member of the committee can see if the same comment has been made by some other member.

This reduces time and effort both by the student as well as the committee members.

3. There will not be any wastage of paper during the review process.
4. There will be no time wasted scheduling meetings, driving to the campus and making special efforts.
5. The reviews and editing can be done by the committee members and students respectively in their own available time. This brings a lot of flexibility to the procedure of proposal review.
6. Athena is customized for Project Proposals only whereas Zeus [2] was designed for Thesis.
7. Athena is follows the IEEE std. [5] for Software Requirement Specifications which will guide students in writing their Proposals either strictly based on the standard or a modified version.
8. Athena is very simple in its implementation.
9. Athena is designed based on the rules and regulations of the Department of Computer Science at CSUSB whereas the other related works like Zeus, Posiedon and Tales are designed for their own University.

10. Athena is for Proposals written in English whereas Zeus and others have been designed for Spanish thesis and documents.
11. The main difference between Athena and Zeus [2] is that in Athena you write or copy and paste the text written In any editor into the Athena Interfaces which is then saved in the database. In Zeus you upload the text in HTML format.
12. In Athena you can always make edits to the document whenever needed, whereas in Zeus [2] you have to load the new file again.
13. Zeus [2] has a very well developed annotation system but Athena has a very simple one.
14. Poseidon [3] is for all kinds of documents but Athena caters only to Project Proposal of Masters Students of Computer Science at CSUSB.
15. Tales [1] is a repository of digital theses whereas this prototype of Athena has not implemented that yet, but it will very easy to add it as well.
16. The communication in all the three related works, Zeus [2], Tales [1], Poseidon [3] is initiated automatically, whereas in Athena the users create

their own emails and send it. The mailing system has been created customized for Athena.

Athena is the first prototype of a system for Collaborative proposal development. Hopefully many others can follow to cover all the scenarios and types of documents and procedures.

6.3 Future Directions

There are innumerable possibilities and features that could be added to Athena. This should be treated and used as a first running prototype of a Proposal Development System. It could be used widely as well as ideas of improving it and making it more generalized could be gathered and improvised in the next prototypes.

A few suggestions are:

1. Beta Testing has to be done on the project by the department before implementing Athena.
2. Athena could be a long term fit to the department web site and help in tracking computer resources.
3. Athena could be made to handle more than 3 chapters. It is not a big programming effort to do it. It was not done in this version because certain limitations were applied to maintain uniformity between proposals.

4. The depth of the levels of sub headings could go to more than 3 levels. Athena utilized it to maintain uniformity and provide some limitations.
5. There can be a better version control system implemented. There is always a current version available in Athena. Old versions are overwritten. But this could change and old versions could be saved.
6. Annotation can be done and shown in layers as implemented in Zeus [2].
7. There could be specific colors for annotations by each committee members.
8. Athena could be generalized for all departments of CSUSB.
9. Athena could be generalized to handle any kind of document, reports, thesis etc.
10. Athena could be upgraded to XML Interfaces or JSP Servlets.
11. On Advancement to Candidacy the proposal could be added to a public directory on Athena or publicized on the department website.
12. Athena could be installed in the Computer Science Website as well the Graduate Advisory System and

utilized for all its worth and made available to all the students of the department.

6.3 Summary

Athena has the capability of becoming a popular and widely used system. It was implemented in a logical way and will be a great asset to the Department of Computer Science. The Advancement to Candidacy procedure will become much easier. Athena would surely help students and faculty work on the proposals with more interest, in a much more relaxed manner, at their own convenience using the user-friendly and clear interfaces. It would bring about a sense of accomplishment and satisfaction as there could be any number of back and forth reviews and annotations without the constraints of time, place, and readability issues. Athena can surely be utilized in the progress of educational system procedures and development of efficient and intelligent proposals by Masters Students of the Department of Computer Science at CSUSB.

APPENDIX A

DATABASE SCHEMA

The database tables used for Athena are shown below; Refer Chapter 2 for a detailed description of the tables and the relationships between these tables/entities.

The table creation scripts used in Athena can be listed as follows:

```
CREATE TABLE users (
user_id INT(4) UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
fname VARCHAR(20) NOT NULL,
lname VARCHAR(20) NOT NULL,
email VARCHAR(50) NOT NULL,
phone VARCHAR(12) NOT NULL,
user_role VARCHAR(10) NOT NULL,
user_name VARCHAR(30) UNIQUE NOT NULL,
password VARCHAR(20) NOT NULL);

CREATE TABLE student_proposal (
prop_id INT(4) UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
stud_id INT(4) UNIQUE NOT NULL REFERENCES users(user_id),
prop_type VARCHAR(4) NULL DEFAULT 'none');

CREATE TABLE proposal (
prop_id INT(4) NOT NULL PRIMARY KEY,
prop_title VARCHAR(100) NOT NULL UNIQUE,
start_date DATE NOT NULL,
end_date DATE NOT NULL);

CREATE TABLE committee_members (
cm_id INT(4) UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY
KEY, prop_id INT(4) NOT NULL REFERENCES proposal(prop_id),
faculty_name VARCHAR(30) NOT NULL REFERENCES users(lname),
role VARCHAR(20) NOT NULL,
comm_status VARCHAR(2) DEFAULT "NO",
review_status VARCHAR(2) DEFAULT "NR");

CREATE TABLE sections_pid (
sec_id INT(4) UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
sec_parent INT(4) NOT NULL,
sec_num INT(4) NOT NULL,
sec_heading VARCHAR(50) NULL,
sec_data TEXT NULL,
sec_image VARCHAR(100) NULL DEFAULT 'none',
sec_icaption VARCHAR(100) NULL DEFAULT 'none');

CREATE TABLE comments_pid(
comment_id INT(4) UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
sec_id INT(4) NOT NULL,
cm_id INT(4) NOT NULL,
comment_faculty VARCHAR(255) NULL,
comment_student VARCHAR(255) NULL);
```

APPENDIX B

SOURCE CODE

These are the source codes for the JavaBeans and JSP pages used in Athena:

Database.java

```
package mpds_modules;
import java.sql.*;
public class database
{
  private static String
url="jdbc:MySql://localhost/mpds";
  private static String
driver="com.MySql.jdbc.Driver";
  private static String user="humaira";
  private static String password="mpds";

  public static Connection getConnection() throws
Exception
  {
    Class.forName(driver).newInstance();
     System.out.println("driver Loaded");
    Connection connection =
DriverManager.getConnection(url,user,password);
     System.out.println("connection established");
    return connection;
  }
}
```

Proposal.java

```
package mpds_modules;
import java.lang.*;
import java.util.*;
import java.sql.*;
public class Proposal
{
     private int pid;
     private String ptitle;
     private String sdate;
     private String edate;

     public Proposal() {  }
     public int getPid() { return this.pid; }
     public String getPtitle() { return this.ptitle;
     }
     public String getSdate() { return this.sdate; }
     public String getEdate() { return this.edate; }
     public void setPid(int p) { pid = p; }
     public void setPtitle(String tit){ ptitle=tit; }
```

```
        public void setSdate(String sd) { sdate=sd; }
        public void setEdate(String ed) { edate=ed; }
    }
```

SectionAdmin.java

```
    package mpds_modules;
    import java.lang.*;
    import java.util.*;
    import java.sql.*;
    import java.net.URLEncoder;
    import mpds_modules.*;

    public class SectionAdmin {
        private Connection connect;
        private int pid;
        private String msg;
        public SectionAdmin() {
            try {
            connect = database.getConnection();
            }catch (Exception ex)    {
                ex.printStackTrace();
            }
        }
    /***************CREATE SECTIONS******************/

        public void createSections(int proposal_id){
            pid = proposal_id;
            try {
            Statement stmt=connect.createStatement();
            String temp = "sections_";
            String new_table = temp +
Integer.toString(pid);

            String update = "create table "+new_table+"
(sec_id INT(4) UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY
KEY,sec_parent INT(4) NOT NULL,sec_num INT(4) NOT
NULL,sec_heading VARCHAR(50));";
            stmt.executeUpdate(update);
            String insert_sec="insert into
"+new_table+" (sec_parent,sec_num,sec_heading) VALUES
(0,1,'Introduction'), (1,1,'Purpose'), (1,2,'Scope'),
(1,3,'Definitions and Acronyms'), (1,4,'References'),
(1,5,'Overview'), (0,2,'Overall Description'),
(7,1,'Product Perspective'), (8,1,'System Interfaces'),
(8,2,'User Interfaces'), (8,3,'Hardware Interfaces'),
(8,4,'Software Interfaces'), (8,5,'Communication
Interfaces'), (8,6,'Memory Constraints'),
```

```
(8,7,'Operations'), (7,2,'Product Functions'), (7,3,'User
Characteristics'), (7,4,'Assumptions and Dependencies'),
(0,3,'Specific Requirements'), (19,1,'Functional
Requirements'), (19,2,'Performance Requirements'),
(19,3,'Logical Database Requirements'), (19,4,'Design
Constraints'), (19,5,'Software System Requirements'),
(24,1,'Relaiblility'), (24,2,'Availability'),
(24,3,'Security'), (24,4,'Maintainability'),
(24,5,'Portability');";
                    stmt.executeUpdate(insert_sec);
                }
     catch(SQLException E) {
          System.out.println("SQLException: " +
E.getMessage());
          System.out.println("SQLState:     " +
E.getSQLState());
          System.out.println("VendorError:  " +
E.getErrorCode());
             } //end catch
          }//createSections
/**************END CREATE SECTIONS**********************/
     /************DELETE SECTIONS BY ID****************/
public String deleteSection(String [] sec_ids, int count1,
String table) {
     try {
     Statement stmt11 = connect.createStatement();
     int t1=0;
     for (int i=0;i<count1;i++) {
     t1 = Integer.parseInt(sec_ids[i]);
     ResultSet RS = stmt11.executeQuery("select
sec_parent,sec_num from "+table+" where sec_id="+t1+" ");
     RS.first();
     int parent = RS.getInt(1);
     int num = RS.getInt(2);
     String query1 =("update "+table+" SET
sec_num=sec_num-1 where sec_parent="+parent+" and
sec_num>"+num+" ");
     stmt11.executeUpdate(query1); //updating section
numbers following this.
     stmt11.executeUpdate("delete from "+table+" where
sec_parent="+t1+" "); //deleting sub-sections
     stmt11.executeUpdate("delete from "+table+" where
sec_id="+t1+" "); //deleting actual section/subsection
               } // end for
} // end try
     catch (SQLException E) {
```

```
     System.out.println("SQLException: " +
E.getMessage());
     System.out.println("SQLState:     " +
E.getSQLState());
     System.out.println("VendorError:  " +
E.getErrorCode());
                    msg=E.getMessage();
} //end catch
          return("done");
} // end deleteSection

/*************END DELETE SECTIONS BY ID**************/
/************INSERT SECTIONS BY ID *********/
     public String insertSection(String [] sec_ids, int
count1, String table) {
     try {
     Statement stmt12 = connect.createStatement();
     int t1=0;
     for (int i=0;i<count1;i++) {
     t1 = Integer.parseInt(sec_ids[i]);
     ResultSet RS = stmt12.executeQuery("select
sec_parent,sec_num from "+table+" where sec_id="+t1+" ");
     RS.first();
     int parent = RS.getInt(1);
     int num = RS.getInt(2);
     String query1 =("update "+table+" SET
sec_num=sec_num+1 where sec_parent="+parent+" and
sec_num>"+num+" ");
     stmt12.executeUpdate(query1); //updating section
numbers following this.
     String temp3 = "<Enter Section Title Here>";
     String query2=("insert into "+table+"
(sec_parent,sec_num,sec_heading) values
("+parent+","+(num+1)+",\""+temp3+"\") ");
     stmt12.executeUpdate(query2); //deleting sub-sections
     } // end for
} // end try
catch (SQLException E) {
     System.out.println("SQLException: " +
E.getMessage());
     System.out.println("SQLState:     " +
E.getSQLState());
     System.out.println("VendorError:  " +
E.getErrorCode());
     msg=E.getMessage();
} //end catch
return("done");
```

```
} // end insertSection

/**********END INSERT SECTIONS BY ID*******************/
/************INSERT SUB-SECTIONS BY ID *********/
     public String insertSubSection(String [] sec_ids, int
     count1, String table) {
     try {
     Statement stmt13 = connect.createStatement();
     int t2=0;
     for (int i=0;i<count1;i++) {
     t2 = Integer.parseInt(sec_ids[i]);
     ResultSet RS = stmt13.executeQuery("select
     sec_parent,sec_num from "+table+" where sec_id="+t2+"
     ");
     RS.first();
     int parent = RS.getInt(1);
     int num = RS.getInt(2);
     String query1 =("update "+table+" SET
     sec_num=sec_num+1 where sec_parent="+t2+" ");
     stmt13.executeUpdate(query1);
     String temp4 = "Enter Sub-Section Title Here>";
     int temp5 = 1;
     String query2=("insert into "+table+"
     (sec_parent,sec_num,sec_heading) values
     ("+t2+","+temp5+",\""+temp4+"\") ");
     stmt13.executeUpdate(query2);
          } // end for
     } // end try
     catch (SQLException E) {
          System.out.println("SQLException: " +
E.getMessage());
          System.out.println("SQLState:     " +
E.getSQLState());
          System.out.println("VendorError:  " +
E.getErrorCode());       msg=E.getMessage();
         } //end catch
          return("done");
     } // end insertSubSection

/**********END INSERT SUB-SECTIONS BY ID ************/
} // END SectionAdmin.java
```

StudentAdmin.java

```
package mpds_modules;
import java.lang.*;
import java.util.*;
import java.sql.*;
import java.net.URLEncoder;
import mpds_modules.*;

public class StudentAdmin {
     public Connection connect;
     private int u_id;
     private int p_id;
     private String p_title;
     private String p_sdate;
     private String p_edate;
     private boolean wrongData;
     private String msg;
     private String ad_name;
     private String cm1_name;
     private String cm2_name;
     private String cm3_name;
     String ad = "advisor";
     String c1 = "cm1";
     String c2 = "cm2";
     String c3 = "cm3";
     public StudentAdmin(){
          try  {
          connect = database.getConnection();
          }catch (Exception ex) {
          ex.printStackTrace();
          }
     }
/*****************ADD
PROPOSAL*****************************/
public String addProposal(Proposal prop1, int uid) {
     u_id=uid;
     p_title=prop1.getPtitle();
     p_sdate=prop1.getSdate();
     p_edate=prop1.getEdate();
     wrongData=false;
     msg=null;
     try  {
          Statement stmt1=connect.createStatement();
          Statement stmt2=connect.createStatement();
          Statement stmt3=connect.createStatement();
```

```
          ResultSet rs1=stmt1.executeQuery("select
prop_id from student_proposal where stud_id="+u_id+"
");
          if(!rs1.next()){
          String insert = "insert into
student_proposal(stud_id) values ("+u_id+");" ;
          stmt1.executeUpdate(insert);
          ResultSet rs33=stmt2.executeQuery("select
prop_id from student_proposal where stud_id="+u_id+"
");
          while(rs33.next()){
          p_id = rs33.getInt(1);
          prop1.setPid(p_id);
          ResultSet rs44=stmt3.executeQuery("select
prop_title from proposal");
          if(rs44.next()) {
          String t=rs44.getString(1);
          if(t.equals(p_title)) {
          wrongData=true;
          }else {
          wrongData=false;
          }
          }else
          {
          wrongData=false;
          }
           if(wrongData){
          msg="A proposal already exists with that
title, enter another one";
          }else{
          String insert1 = "INSERT INTO
proposal(prop_id, prop_title, start_date, end_date)
VALUES ("+p_id+", \""+
          p_title+"\", \""+p_sdate+"\",
\""+p_edate+"\"); " ;
          stmt3.executeUpdate(insert1);
     //create the sections table
SectionAdmin sections=new SectionAdmin();
sections.createSections(p_id);
//sent to the SectionsAdmin class
msg="done";
stmt3.close();
}//else wrongdata
 }//while
rs33.close();
stmt2.close();
}else{
```

```
System.out.println("we are in the same proposal
loop");
msg="There cannot be two proposals for the same user.
";
}//else
rs1.close();
stmt1.close();
connect.close();
}catch(SQLException E) {
System.out.println("SQLException: " +
E.getMessage());
System.out.println("SQLState:     " +
E.getSQLState());
System.out.println("VendorError:  " +
E.getErrorCode());
} //end catch
return(msg);
 }//addProposal

/**************END ADD PROPOSAL*****************/
/*************MODIFY PROPOSAL*******************/
public String modifyProposal(Proposal pi, int pid) {
p_id=pid;
pi.setPid(p_id);
p_title=pi.getPtitle();
p_sdate=pi.getSdate();
p_edate=pi.getEdate();
msg=null;
try  {
System.out.println("I am in modify");
Statement stmt45 = connect.createStatement();
String update = "UPDATE proposal SET prop_title=\""+
p_title +"\", start_date= \""+ p_sdate +"\",
end_date=\""+ p_edate +"\" where prop_id="+p_id+" ";
stmt45.executeUpdate(update);
System.out.println("I am in modify after update");
msg="done";
}catch(SQLException E) {
System.out.println("SQLException: " +
E.getMessage());
System.out.println("SQLState:     " +
E.getSQLState());
System.out.println("VendorError:  " +
E.getErrorCode());
msg=E.getMessage();
} //end catch
return(msg);
```

```
}//modifyProposal

/*****************MODIFY
PROPOSAL**********************/
/*****************CREATE COMMITTEE
*********************/
public String createCommittee(String [] com, int
p_id){
ad_name=com[0];
cm1_name=com[1];
cm2_name=com[2];
cm3_name=com[3];
msg=null;
wrongData=false;
try {
if (ad_name.equals(cm1_name) ||
ad_name.equals(cm2_name) ||
ad_name.equals(cm3_name)){
msg="Duplicate Entries matching with Advisor.  Choose
new again.\n";
wrongData=true;
}else{
if(cm1_name.equals(cm2_name)||
cm1_name.equals(cm3_name)){
msg="Duplicate Entries matching with Committee
Members.  Choose new again.\n";
wrongData=true;
}else{
if(cm2_name.equals(cm3_name)){
msg="Duplicate Entries matching with Committee
Members.  Choose new again.\n";
wrongData=true;
}else{
wrongData=false;
}//else cm2-cm3
wrongData=false;
}//elsecm1
wrongData=false;
}//elseadvisor
Statement stmt10 = connect.createStatement();
if(!wrongData){
String insert_ad="insert into committee_members
(prop_id, faculty_name, role) values ("+p_id+",\""
+ad_name+"\", \""+ad+"\");" ;
String insert_c1="insert into committee_members
(prop_id, faculty_name, role) values ("+p_id+",\""
+cm1_name+"\", \""+c1+"\");" ;
```

```
String insert_c2="insert into committee_members
(prop_id, faculty_name, role) values ("+p_id+",\""
+cm2_name+"\", \""+c2+"\");" ;
stmt10.executeUpdate(insert_ad);
stmt10.executeUpdate(insert_c1);
stmt10.executeUpdate(insert_c2);
msg="done";
stmt10.close();
connect.close();
}else
{
msg="Duplicate Entries matching with Committee
Members.  Choose new again.\n";
}//elsewrongData
 }catch (SQLException E) {
System.out.println("SQLException: " +
E.getMessage());
System.out.println("SQLState:     " +
E.getSQLState());
System.out.println("VendorError:  " +
E.getErrorCode());
msg=E.getMessage();
 } //end catch
 return(msg);
}//createCommittee
/**************CREATE COMMITTEE
*************************/
/*****************UPDATE
COMMITTEE**********************/
public String updateCommittee(String [] com, int
p_id)
{
 ad_name=com[0];
 cm1_name=com[1];
 cm2_name=com[2];
 cm3_name=com[3];
 msg=null;
 wrongData=false;
 try {
if (ad_name.equals(cm1_name) ||
ad_name.equals(cm2_name) ||
ad_name.equals(cm3_name)){
wrongData=true;
}else{
if(cm1_name.equals(cm2_name)||
cm1_name.equals(cm3_name)){
wrongData=true;
```

```
}else{
if(cm2_name.equals(cm3_name)) {
wrongData=true;
}else{
wrongData=false;
}//else cm2-cm3
}//elsecm1
}//elseadvisor
Statement stmt10 = connect.createStatement();
if(!wrongData){
String update_ad="update committee_members set
faculty_name=\""+ad_name+"\" where prop_id="+p_id+"
&& role='"+ad+"' ";
String update_c1="update committee_members set
faculty_name=\""+cm1_name+"\" where prop_id="+p_id+"
&& role='"+c1+"' ";
String update_c2="update committee_members set
faculty_name=\""+cm2_name+"\" where prop_id="+p_id+"
&& role='"+c2+"' ";
String update_c3="update committee_members set
faculty_name=\""+cm3_name+"\" where prop_id="+p_id+"
&& role='"+c3+"' ";
stmt10.executeUpdate(update_ad);
stmt10.executeUpdate(update_c1);
stmt10.executeUpdate(update_c2);
stmt10.executeUpdate(update_c3);
msg="done";
stmt10.close();
connect.close();
}else {
msg="Duplicate Entries matching with Committee
Members.  Choose new again.\n";
}//elsewrongData
 }catch (SQLException E) {
System.out.println("SQLException: " +
E.getMessage());
System.out.println("SQLState:     " +
E.getSQLState());
System.out.println("VendorError:  " +
E.getErrorCode());
msg=E.getMessage();
} //end catch
return(msg);
}//updateCommittee
/*************END UPDATE COMMITTEE
*******************/
}//class
```

Users.java

```
     package mpds_modules;
     import java.lang.*;
     import java.util.*;
     import java.sql.*;
     public class Users {
     private int user_id;
     private String fname,lname,email,user_role,user_name;
     private String password;
     public Users(){     }
     public int getUserid()    { return this.user_id; };
     public String getFname() { return this.fname; }
     public String getLname() { return this.lname; };
     public String getEmail() { return this.email; }
     public String getUserrole() { return this.user_role;}
     public String getUsername() { return this.user_name;}
     public String getPassword() { return this.password;}
     public void setUserid(int userid){
this.user_id=userid;}
     public void setFname(String f_name)
     {this.fname=f_name;}
     public void setLname(String
l_name){this.lname=l_name;}
     public void setEmail(String
email_ad){this.email=email_ad;}
     public void setUserrole(String userrole){
     this.user_role=userrole;}
     public void setUsername(String username){
     this.user_name=username; }
     public void setPassword(String pass){
this.password=pass; }
     }//class
```

UsersAdmin.java

```
        package mpds_modules;
        import java.lang.*;
        import java.util.*;
        import java.sql.*;
        import java.net.URLEncoder;
        import mpds_modules.*;

        public class UsersAdmin{
           public Connection connect;
           private String msg;
           private String ID;
           private String First;
           private String Last;
```

```
    private String Email;
    private String Role;
    private String Name;
    private String Pass;
    boolean wrongData;

    public UsersAdmin(){
         try  {
         connect = database.getConnection();
         }catch (Exception ex){
         ex.printStackTrace();
         }
    }

    public String createUser(Users u) {
    wrongData = false;
    msg=null;
    First=u.getFname();
    Last=u.getLname();
    Email=u.getEmail();
    Role=u.getUserrole();
    Name=u.getUsername();
    Pass=u.getPassword();
    try{
    Statement stmt=connect.createStatement();
    Statement stmt2=connect.createStatement();
    ResultSet rs = stmt.executeQuery("select
user_name from users");
    while(rs.next())    {
    String dbuname = rs.getString(1);
    if(dbuname.equals(Name)){
    wrongData = true;
    }//if userid
    else{
    wrongData = false;
    }//else userid
    }//while
    rs.close();
    if(wrongData){
    msg="Enter another username as this one already
    exists!!";
    }//if wrongData
    else{
    String update="INSERT INTO
    users(fname,lname,email,user_role,user_name,pass
    word) VALUES (\"" +First + "\", \"" +Last  +
    "\", \"" +
```

```
   Email + "\", \"" +Role  + "\", \"" +Name  + "\",
   \"" +   Pass  + "\"); ";
   stmt2.executeUpdate(update);
   msg="done";
   connect.close();
   }
   }catch(SQLException E) {
    System.out.println("SQLException: " +
E.getMessage());
    System.out.println("SQLState:     " +
E.getSQLState());
    System.out.println("VendorError:  " +
E.getErrorCode());
   } //end catch
   return(msg);
   }//createUser

   public String deleteUser(String uname) {
     try     {
     Statement stmt=connect.createStatement();
     String delete = "delete from users where
      user_name='"+uname+"';";
     stmt.executeUpdate(delete);
     msg="done";
     stmt.close();
     connect.close();
     }catch(SQLException E) {
      System.out.println("SQLException: " +
E.getMessage());
      System.out.println("SQLState:     " +
E.getSQLState());
    System.out.println("VendorError:  " +
E.getErrorCode());
   } //end catch
   return(msg);
   }//deleteUser

   public String editUser(Users u) {
   wrongData = false;
   msg=null;
    First=u.getFname();
   Last=u.getLname();
   Email=u.getEmail();
   Role=u.getUserrole();
   Name=u.getUsername();
   Pass=u.getPassword();
   try {
```

```
            Statement stmt=connect.createStatement();
            String update = "UPDATE users set fname = \"" +
          First +"\" , lname = \"" + Last +"\" , email = \""
          + Email +"\" , user_role = \"" + Role + "\" ,
          user_name = \"" + Name +"\" , password =
          \""+Pass+"\" where user_name='"+Name+"' ";
            stmt.executeUpdate(update);
            msg="done";
            stmt.close();
            connect.close();
            }catch(SQLException E) {
             System.out.println("SQLException: " +
          E.getMessage());
             System.out.println("SQLState:     " +
          E.getSQLState());
             System.out.println("VendorError:  " +
          E.getErrorCode());
             } //end catchreturn(msg);
            }//editUser
       public String loginUser(String user,String
role,String pswd) {
          Name=user;
          Role=role;pass=pswd;
          try {
            Statement stmt=connect.createStatement();
            ResultSet rs = stmt.executeQuery("select * from
          users");
            System.out.println("I am here");
            while(rs.next()){
            String dbuser = rs.getString(6);
            if(dbuser.equals(Name))  {
            String dbpswd = rs.getString(7);
            if(dbpswd.equals(Pass)) {
            String rolename = rs.getString(5);
            if (rolename.equals(Role)){
            System.out.println("I am here inside the loop");
            msg="done";
            break;
            }else{
            msg = "Role Name does not match the existing
            one. Please try again or Contact the System
            Administrator!!";
            }//elseRole
            }else{
            msg = "Password doesn't match. Please try again
            or Contact the System Administrator!!";
            }//elsePassword
```

```
   }else{
   msg = "User Name is not in database. Create User
Account first by contacting the System
Administrator!!";
   }//elseUsername
   }//while
   rs.close();
   stmt.close();
   connect.close();
   }catch(SQLException E) {
    System.out.println("SQLException: " +
E.getMessage());
    System.out.println("SQLState:     " +
E.getSQLState());
    System.out.println("VendorError:  " +
E.getErrorCode());
    } //end catch
   return(msg);
   //loginUser
   }//UsersAdmin class
```

REFERENCES

[1] Fernandez L., Sanchez J.A., Community Tales: An Infrastructure for the collaborative construction of digital theses repositories. Proceedings of the Sixth International Conference on Electronic Theses and Dissertations(ETD 2003, Berlin, Germany, May).

[2] Fernandez L., Sanchez J.A., Flores A., An environment for the collaborative revision of digital theses. Proc. Of the 6th Intl. Workshop on Groupware (CRIWG 2000, Madeira, Portugal), IEEE Computer Society Press, 150-153, 2000.

[3] Sanchez J.A., Flores A., Provisions for collaborative revision and annotation of digital documents, Conference on Computer Supported Cooperative Work, ACM Press 2002.

[4] Girgensohn A., Lee A., Schluter K., Experience in developing collaborative applications using the world wide web "Shell", The Seventh ACM Conference on Hypertext, Washingtin D.C., March 1996.d

[5] IEEE Std. 830-1998, IEEE Recommended Practice for Software Reqirement Specification - Annex A.

[6] Flower and Scott, UML Distilled, A Brief Guide to Standard Object Modelling Language, 1999.

[7] Perry D.E., Porter A., Wade M.W., Votta L.G., Perpich J., Reducing Inspection Interval in Large-Scale Software Development, IEEE Transactions on Software Engineering, Vol 28, No. 7, July 2002.

[8] Naughton P.,Schildt H., Java 2 - The Complete Reference, Tata McGraw Hill 1999.

[9] Tremblett P., Instant JavaServer Pages, McGraw Hill 2000.

[10] Tulder G.V., PHP and MySql Tutorials, Storing Heirarchical data in a Database, http://www.sitepoint.com/articles/heirarchical-data-database, April 2003.